МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Фоменко Станіслав Олегович

Програмне забезпечення системи моніторингу продуктивності мережі на базі SNMP/NMS

Спеціальність: 123 «Комп'ютерна інженерія»

Освітній ступінь: бакалавр

Науковий керівник:

Дрєєв Олександр Миколайович _____

(підпис) (дата)

кандидат технічних наук, доцент

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *123 Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« 11 » січня 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Фоменку Станіславу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи моніторингу продуктивності мережі на базі SNMP/NMS*

керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 204-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи моніторингу продуктивності мережі на базі SNMP/NMS*

4. 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

5. 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*

6. Дата видачі завдання « 11 » січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2021 р.	
3.	Розробка моделі компонента	20.03.2021 р.	
4.	Розробка структур даних	25.03.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2021 р.	
6.	Програмування алгоритмів	10.04.2021 р.	
7.	Оформлення ПЗ	17.04.2021 р.	
8.	Попередній захист роботи	14.05.2021 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Фоменко С.О. Програмне забезпечення системи моніторингу продуктивності мережі на базі SNMP/NMS. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи моніторингу продуктивності мережі на базі SNMP/NMS.

Метою розробки є програмне забезпечення системи моніторингу продуктивності мережі на базі SNMP/NMS.

Результат роботи – програмна реалізація системи моніторингу продуктивності мережі на базі SNMP/NMS.

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 XP/Vista/7/8/10.

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерна інженерія, моніторинг, SNMP, NMS

ABSTRACT

Fomenko SO SNMP / NMS based network performance monitoring system software. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this bachelor's qualification the software which is intended for system of monitoring of network productivity on the basis of SNMP / NMS is developed.

The purpose of the development is the software of the network performance monitoring system based on SNMP / NMS.

The result is a software implementation of a network performance monitoring system based on SNMP / NMS.

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.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in the Builder C ++ environment.

Keywords: computer engineering, monitoring, SNMP, NMS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	–	Програмне забезпечення
IETF	–	Інженерна група Інтернету
IT	–	Інформаційні технології
MIB	–	База керуючої інформації
NMS	–	Система керування мережною інфраструктурою
NPM	–	Моніторинг продуктивності мережі
OID	–	Ідентифікатор об'єкта
PDU	–	Protocol Data Unit
PVI	–	Protocol-version independent
RMON MIB	–	Об'єкти MIB віддаленого мережного моніторингу
SLA	–	Угода про рівень обслуговування
SNMP	–	Simple Network Management Protocol
VACM	–	View-based Access Control Model
USM	–	User-based Security Model

ВСТУП

Актуальність теми. Корпоративні замовники, які приходять до нас із пілотними й уже робочими проектами, при виборі застосунків для моніторингу продуктивності мережі (Network Performance Monitoring, NPM) звичайно намагаються розв'язати наступні завдання:

– забезпечення ефективного й діючого контролю над роботою IT-інфраструктури, як єдиного середовища по доставці застосунків і сервісів до користувачів, а не розрізнених елементів, кожний з яких почуває себе відмінно;

– зниження загальної вартості володіння по експлуатації й ефективне вкладення грошей у модернізацію IT-інфраструктури.

Необхідно розуміти, що застосунки для NPM призначені для виявлення й надання докладного звіту про виниклу проблему й стани різних елементів IT-інфраструктури, які задіяні в доставлянні цього сервісу. Таким чином, IT-фахівці потрібного профілю здатні відразу приступити до вирішення завдання й не витрачати час на додаткові установки аналізаторів трафіку, Flow потоків або SNMP.

Крім озвучених цілей для IT-фахівців, застосунки для моніторингу й аналізу продуктивності мережі повинні задовольняти вимогам бізнесу. Бізнесу важливо бачити інформацію про якісні показники стану сервісів продуктивності, що й постійно поліпшується, IT-інфраструктури.

Таким чином, застосунки треба розглядати в розрізі прийнятих у компанії бізнес процесів. Наприклад, ви розв'язали внести зміни в конфігурацію мережі або закупити нове обладнання й система NPM на основі щотижневих звітів повинна показати якісні зміни в роботі сервісів. Якщо їх не буде, то гроші витрачені даремно. Щомісячні й щоквартальні звіти дозволять вчасно контролювати використання і ємність IT-інфраструктури. І якщо використання її

повністю відповідає бізнесу, а продуктивність низька, те вчасно запланувати бюджет на модернізацію відомих елементів IT-інфраструктури.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи моніторингу продуктивності мережі на базі SNMP/NMS.

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

– Огляд існуючих систем моніторингу продуктивності мережі на базі SNMP/NMS.

– Дослідження системи моніторингу продуктивності мережі на базі SNMP/NMS.

– Програмна реалізація системи моніторингу продуктивності мережі на базі SNMP/NMS.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моніторингу продуктивності мережі на базі SNMP/NMS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи моніторингу продуктивності мережі на базі SNMP/NMS, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Мережа – це серце IT-інфраструктури. Коли мережа перестає працювати, потік інформації, необхідний застосункум і бізнес-операціям, зупиняється.

Мережних адміністраторів постійно просять додати нових користувачів і технологічні застосунки в мережі компанії. Ці зміни можуть вплинути на їхню здатність забезпечувати надійну й передбачувану продуктивність мережі. IT-персонал і мережні адміністратори змушено шукати причини проблеми до того, як вона торкнеться користувачів, технологій і бізнес-застосунків. Це складніше, коли виникають проблеми із продуктивністю, які складно відтворити й проаналізувати.

Оцінка проблем із продуктивністю допомагає IT-командам знаходити проблеми на попередні етапах. Ефективна система моніторингу мережі допомагає в запобіганні збоїв мережі або простоїв.

Згодом моніторинг мережі значно розширився, що привело до система моніторингу мережі автоматизації. Крім того, є введення в кілька способів відстеження систем і серверів, оповіщення зацікавлених сторін.

У галузі з'явилися нові модернізовані способи підключення за допомогою таких технологій, як віддалені користувачі, мобільні пристрої, бездротові мережі, хмарні технології, IoT, VPN і багато інші. Крім того, моніторинг мережі став необхідним для моніторингу більш досконалих мереж, таких як LAN, MPLS, VoIP, WAN і т. Д.

SLA – це договір, підписаний між IT-фахівцями й власниками бізнесу. Плани компенсації IT призначені для виконання зобов'язань SLA, щоб гарантувати або гарантувати рівень продуктивності. Вони відслідковуються, виміряються й періодично повідомляються. Керування SLA Важливо те, що

погана продуктивність і простої в наші дні досить дорогі. Для популярного сайту електронної комерції хвилини простою від 30 до 45 можуть привести до мільйонів загублених доходів.

Вимоги SLA можуть бути надзвичайно строгими. Деякі провідні компанії вимагають 99.99% часу безвідмовної роботи. Це дозволяє менш однієї години простою щороку.

У міру того як очікування або вимоги SLA зростають, тем дорожче стає його впровадження, підтримка й підтримка.

Моніторинг мережі починається із процедури виявлення. Він виявляє, що все присутнє і підключений до мережі для моніторингу. Моніторинг продуктивності мережі (NMS) досліджує пристрої в мережі, які включають сервери, маршрутизатори, комутатори, брандмауери, принтери й багато чого іншого.

NMS містить бібліотеку шаблонів моніторингу, у якій вказується, як відслідковувати пристрій. Коли NMS завершує процес виявлення, вона автоматично призначає підходящу роль пристрою кожному виявленому пристрою.

Здатність мережного адміністратора візуалізувати свої мережі може допомогти заощадити час і зусилля по усуненню неполадок у мережі. Однак масштабованість мережі обмежує можливості мережних адміністраторів візуалізувати мережу й стримувати вирішення проблем.

Мережні карти є впливовим інструментом реагування, який дозволяє мережним адміністраторам навіть візуалізувати складні мережі. Карта мережі демонструє пристрою і їх останній статус.

Системи моніторингу мережі попереджають адміністраторів мережі, коли пристрою виходять із ладу. Вони доставляють повідомлення по електронній пошті, текстам і шляхом реєстрації. Одержання оповіщень про задані граничні значення важливо для будь-якого мережного адміністратора, щоб він почав правильні дії в потрібний час.

Системи мережного моніторингу можуть генерувати попередження й повідомлення, а також можуть інтегруватися зі сторонніми застосункуми для більш плавної роботи.

1.2 Область застосування

NMS пропонує ролі пристроїв під ключ, які визначають, що відслідковувати або відслідковувати. Мережні адміністратори можуть адаптуватися, щоб придумувати ролі або створювати нові. NMS надають мережним адміністраторам величезну колекцію моніторів.

Стежите за основними процесами, які займають найбільше місця на вашому процесорі. Ознайомтеся з основними показниками, що впливають на загальну продуктивність мережі.

Мережні адміністратори зайняті в постійному життєвому циклі проектування, аналізу й перепроектування мереж. Щоб підтримати цей життєвий цикл, системи NMS пропонують у реальному часі й історичні дані відстеження. Ці дані дають можливість адміністраторам мережі:

Застосунки для моніторингу мережі

Процес мережного моніторингу спрощується й автоматизує за допомогою програмного забезпечення й інструментів для мережного моніторингу. Ці інструменти необхідні для усунення вузьких місць у мережі й проблем, що впливають на продуктивність мережі.

Функціональні можливості ефективних програмних засобів моніторингу мережі:

- Візуалізуйте свою повну IT-інфраструктуру.
- Автоматичне налаштовування пристроїв і інтерфейсів.
- Відстеження продуктивності сервера, мережі й застосунків.
- Усунути несправності мережі, з'ясувавши причину проблеми.

Ви також одержуєте модернізовані функції звітності із плануванням і автоматизацією. Подивитеся на Переваги застосунку для моніторингу продуктивності мережі.

Переваги застосунку для моніторингу мережі містять у собі:

1. Випереджати час простою IT

Простої IT-устаткування відбуваються через людські помилки, проблеми, пов'язані з конфігурацією, і інших природніх факторів. Розгорнувши систему моніторингу мережі, ви можете захистити свою IT-інфраструктуру від цих простоїв і їх виникнення. Крім того, програмний застосунок для моніторингу забезпечує необхідну прозорість, щоб IT-адміністратори могли залишатися в курсі можливих мережних проблем. Відображення працездатності мережі або доступності даних у реальному часі – простий спосіб зрозуміти графічний інтерфейс. Застосунки з мережного моніторингу допоможуть вам відразу ж виявити збої, які можуть викликати проблеми із простоєм IT.

2. Виявити й швидко розв'язати проблеми

Під час простою час і зусилля витрачаються даремно. І прийшов час, щоб ваш застосунок для моніторингу мережі прискорювало й спрощувало вирішення проблем для IT-фахівців вашої компанії. Будь те нерегулярний мережний трафік або збій конфігурації, застосунок для моніторингу мережі допомагає IT-фахівцям вирішувати проблеми, знаходячи аналіз першопричин. Топологія мережі в реальному часі пропонує необхідну інформацію про джерело проблеми або проблеми й надає оновлену інформацію про продуктивність мережних пристроїв у режимі реального часу. Крім того, інструменти мережного моніторингу допомагають автоматично вирішувати проблеми без ручного втручання.

3. Досягніться миттєвого ROI

IT-фахівці з мережних технологій зустрічаються з величезними робочими навантаженнями й зустрічаються з багатогранними мережними проблемами знову й знову без точного часу, витрат і ресурсів, необхідних для їхнього застосунки. Правильна система керування мережею може забезпечити миттєву

окупність інвестицій без необхідності вручну вирішувати проблеми й дозволяючи персоналу працювати над більш значними проектами.

4. Зменшити погрози кібербезпеки

Інструмент мережного моніторингу дозволяє вам боротися з витоками даних для забезпечення безпеки ваших важливих бізнес-даних. Максимальна перевага, яку ви одержите від цього, – це ясна картина того, як «оптимальна» продуктивність мережі виглядає для інфраструктури вашої компанії. Ця прозорість спрощує виявлення всього недовіру, що виникає у вашій мережі.

5. Готові до дозрівання мереж

У міру розвитку технологій і бізнесу навіть мережі ростуть у розмірах. Сучасне устаткування, застосунки й усі пристрої повинні постійно відслідковуватися й контролюватися для будь-яких додаткових дій.

Рух уперед

Ціль інструментів моніторингу мережі полягає не тільки в тому, щоб повідомити IT-адміністраторів про проблеми в мережі, але й у тому, щоб продемонструвати тенденції в продуктивності IT-мережі, допомагаючи їм відповідати певним вимогам. SLAs, Система керування мережею, якщо її використовувати стратегічно й грамотно, допоможе вам подолати всі виникаючі технічні уразливості.

Проте, це може бути важким завданням, щоб повністю переконати ваших старших за один раз, що сервер вимагає відновлення без пояснення всеосяжної логіки. Однак історичний звіт про те, як ця система працювала протягом попередніх декількох місяців, набагато більш переконливий, щоб одержати необхідне твердження бюджету.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи моніторингу продуктивності мережі на базі SNMP/NMS, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Безкоштовні програми для моніторингу мережі в Windows. За допомогою даних засобів, користувач зможе об'єктивно оцінити пропускну здатність підключення, виміряти швидкість підключення в будь-якій мережі. Інші інструменти дозволяють пакети TCP / IP, що проходять через мережний адаптер, відслідковувати мережні компоненти й багато іншого.

The Dude 6.48.1

The Dude – безкоштовна програма моніторингу мережі, у програмі реалізоване застосування нових «Mikrotik» які можуть значно поліпшити ваші шляхи, управляти вашим мережним середовищем. Dude автоматично сканує всі пристрої в зазначених підмережах, малює макет вашої карти мережі, проводить моніторинг служб Вашого пристрою й попереджає Вас у випадку, якщо в деяких служб є проблеми.

PRTG Network Monitor 21.1.65

PRTG Network Monitor – мережний монітор, потужний застосунок мережного моніторингу від компанії Paessler AG. Програма забезпечує доступність мережних компонентів, а також заходу зі спостереженням і їх використанням. Це заощаджує витрати, уникаючи простоїв, оптимізації з'єднання, заощаджуючи час і контроль угод про рівень обслуговування (SLA). PRTG Network Monitor наступне покоління інструментів моніторингу, який комбінує весь досвід компанії мережного моніторингу Paessler із сучасним AJAX сервісного інтерфейсу й новим сучасним движком моніторингу, який призначений для мереж будь-якого розміру.

Nmap 7.91

Nmap – є вільним і відкритим вихідним кодом (безкоштовна ліцензія) утиліта для розвідки мережі або аудита безпеки. Багато мережних адміністраторів також можуть знайти його корисним для таких завдань, як Network Inventory, керування графіком, підвищення класу обслуговування, а також моніторинг хосту. Nmap використовує неопрацьовані пакети IP у нових способах визначення, які хости доступні в мережі, які послуги (ім'я застосунку й версію) ці вузли пропонують, під якими операційними системами (і версіями ОС) вони працюють, який тип пакета фільтри / брандмауери перебувають у використанні, і десятки інших характеристик.

Networx 5.5.5 + Portable

Networx – є простим і безкоштовним, але потужний інструмент, який дозволяє об'єктивно оцінити пропускну здатність вашого підключення. Ви можете використовувати його, щоб збирати дані про пропускну здатність підключення й виміряти швидкість вашого інтернету або будь-якого іншого мережного підключення. Networx може допомогти вам визначити можливі джерела проблем у мережі, переконаєтеся, що ви не перевищуєте межі пропускного каналу, отриманим від вашого провайдера, або можете відслідковувати підозрілі характерні ознаки мережної активності троянських коней і атак хакерів.

Adapterwatch

AdapterWatch – проста безкоштовна програма, яка здійснює моніторинг усіх доступних мережних адаптерів. Для кожного виявленого мережного адаптера програма відображає масу корисної інформації: IP адреса, апаратна адреса, WINS-сервер, DNS-сервер, максимальний розмір пакета (MTU), кількість прийнятих і переданих байт швидкість, що тече, передачі даних і багато чого іншого. Крім іншого, AdapterWatch відображає загальну статистику TCP, IP, UDP, ICMP на локальному комп'ютері.

Не вимагає установки. Працює під керуванням операційних систем Microsoft Windows 2000, 2003, 2008, XP, Vista, 7, 8, 8.1 і 10.

Network Activity Indicator

Network Activity Indicator – невелика безкоштовна програма, яка являє собою альтернативний індикатор мережної активності для операційних систем Microsoft Windows 7, 8, 8.1 і 10 у стилі XP (миготлива синім кольором іконка "два монітори") з відповідною інформативністю й функціональністю.

Після запуску програма займає звичне місце в системному треї, відображаючи передачу вхідного й вихідного трафіку. Показує мережну активність як усіх активних мережних адаптерів, так і обраних користувачем, для чого в налаштовуваннях треба вказати конкретний адаптер і потрібний тип трафіку.

Також програма може виводити статистичну інформацію про роботу мережі, надавати доступ до налаштовувань вбудованого Брандмауера Windows і виконувати інші дії.

Netresview

Netresview – невелика безкоштовна програма, яка відображає список усіх мережних ресурсів (комп'ютери, загальнодоступні диски й принтера) у локальній мережі.

Дана програма показує мережні ресурси із усіх доменів (робочих груп) в одному вікні, у тому числі й сховані.

Не вимагає установки. Працює під керуванням операційних систем Microsoft Windows 2000, 2003, 2008, XP, Vista, 7, 8, 8.1 і 10.

NetRouteView

NetRouteView – невелика безкоштовна програма, що є альтернативою стандартній утиліті маршрутизації (route.exe) операційної системи Microsoft Windows.

Дана програма відображає список усіх маршрутів у мережі, включаючи маску, шлюз, інтерфейс на IP-адреса, дані метрики, тип, протокол, MAC-адреса й

інше. Плюс до всього, NetRouteView дозволяє легко додавати, видаляти або змінювати маршрути.

Програма проста у використанні й не вимагає установки.

Speedtest by Ookla

Speedtest by Ookla – невелика, безкоштовна й проста у використанні десктопная програма для Microsoft Windows, яка пропонує користувачеві найшвидший і легкий спосіб перевірити швидкість інтернет-підключення. Програма дуже проста у використанні, і не містить ніяких налаштовувань. Для початку виміру необхідно просто нажати кнопку “Почати”.

Дана програма дозволяє виміряти швидкість вхідного й вихідного з'єднання, пінг, джиттер (пульсації) і втрату пакетів. Крім того, у програмі присутня можливість подивитися історію вимірів, а саме скільки всього тестів уже запускали, середнє значення швидкості Інтернету й підсумки тесту найшвидшого з'єднання.

Також можна поділитися результатом, відправивши друзям і знайомим посилання на підсумки вимірів.

Getscreen.me

Getscreen.me – хмарний застосунок для дистанційного доступу до робочого стола, що дозволяє управляти користувацькими комп'ютерами й серверами, підключаючи цілі офіси й підприємства.

Відмітною рисою даного інструмента є робота на основі веб-технологій (Webrtc), що дозволяє підключатися до пристроїв по звичайному посиланню прямо з веб-браузера.

Вам більше не буде потрібно обмінюватися ідентифікаторами й паролями, а для одноразового підключення не потрібно навіть установки, завдяки чому процес з'єднання спрощується просто максимально.

Особистий кабінет адаптований під мобільні пристрої з використанням технології Progressive Web Application рятує від нестатку завантажувати

додаткові застосунки, досить просто зберегти сторінку особистого кабінету на домашній екран смартфону.

Підтримує роботу в операційних системах Microsoft Windows, MAC OS X і Linux.

Основні можливості Getscreen.me:

- Доступ з веб-браузера.
- Не вимагає установки для одноразових підключень.
- Підключення без додаткових паролів і посилань.
- Обмін файлами в обидва боки.
- Буфер обміну.
- Зручна робота з декількома моніторами у вкладках веб-браузера.
- Чати й дзвінки усередині сесій.
- Особистий кабінет адаптований під мобільні пристрої.
- Єдиний центр керування для всіх пристроїв.
- Історія сесій.
- Делегування доступу.
- Peer-to-peer підключення.
- Незалежність від виділених IP-адрес.
- І інші майбутні поліпшення.

Nickware Netchecker

Nickware Netchecker – корисна безкоштовна програма для індикації стану підключення до мережі Інтернет у реальному часі, а також для одержання докладної інформації про це підключення.

Підтримує роботу в операційних системах Microsoft Windows XP, Vista, 7, 8 і 8.1.

Особливості Nickware Netchecker:

- Спостереження за станом підключення комп'ютера до мережі Інтернет у реальному часі.
- Індикація при збоях у роботі мережі й звукові оповіщення.

– Визначення неполадок у роботі мережі.

– Можливість перегляду інформації про трафік (поточна швидкість обміну даними, обсяг переданих / отриманих даних).

– Можливість перегляду статистики за весь час, або за певний період.

– Не вимагає установки.

Xirrus Wi-Fi Inspector

Xirrus Wi-Fi Inspector – корисна програма для виявлення Wi-Fi мереж і надання по них різноманітної інформації, як наприклад потужність сигналу, тип мережі, виробника роутера, канал передачі й багато чого іншого.

Пошук хот-спотів відображається в наочному виді за допомогою невеликого радара, який крім усього дозволяє оцінити далекість від вас джерела сигналу.

Крім іншого, програма Xirrus Wi-Fi Inspector надає детальну інформацію про мережу, до якої ви в даний момент підключені, включаючи персональний IP-адресу, зовнішній IP-адреса, DNS, інформацію про шлюз і ін., а також дає можливість протестувати швидкість і якість з'єднання.

Програма безкоштовна й дуже проста у використанні. Працює під керуванням операційних систем Microsoft Windows і MAC OS X.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++ , тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об’єму в порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об’єктно-орієнтований підхід дає

можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об’єктами вашої програми. Окрім типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

Builder C++ підтримує останні розширення стандарту мови C++ та забезпечує швидку компіляцію та складання 32-розрядних програм для Windows. Результуючі програми оптимізовані з точки зору швидкості виконання програм та затрат пам’яті. Зручний відладгоджувальник (з асемблерним вікном, можливістю крокового виконання, завдання точок зупинки, трасування та інше) повністю інтегрований у систему проектування. Дизайнер форм, редактор коду, інспектор об’єктів та інші інструменти зостаються доступними під час виконання програми, саме через це вносити зміни до коду можна прямо у процесі відлагодження.

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі відлагодження.

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve.

Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

Довідкова служба Builder C++ надавала мені допомогу в цій і багатьох інших подібних ситуаціях. Є повний опис кожного управляємого компонента, включаючи списки властивостей і методів, а також численні приклади. Виклад матеріалу в книзі був значно покращуваний і систематизований завдяки відомостям, почерпнутим мною з довідкової служби.

Завдяки засобам управління проектами, двосторонній інтеграції застосунку і синхронізації між засобами візуального і текстового редагування, а також вбудованому відладнику (з асемблерним вікном прокрутки, покрокового виконання, точок останову, трасуванням і тому подібне) – Builder C++ корпорації Borland надає собою вражаюче середовище розробки, яка, мабуть, витримає конкурентну боротьбу з такими модними продуктами як Developer Studio фірми Microsoft.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи моніторингу продуктивності мережі на базі SNMP/NMS.

В процесі розробки кваліфікаційної бакалаврської роботи 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

Simple Network Management Protocol (SNMP) – це протокол прикладного рівня, він уможливлює обмін даними між мережними пристроями.

SNMP – це не продукт, а звід правил. Він визначений Радою по архітектурі Інтернету і є частиною пакета TCP/IP. SNMP управляється й підтримується Інженерною групою Інтернету (IETF).

Протокол дозволяє системному адміністраторові проводити моніторинг, контролювати продуктивність мережі й змінювати конфігурацію підключених пристроїв. SNMP використовують у мережах будь-якого розміру: чому крупніше мережа, тем краще розкриваються переваги протоколу. Він дозволяє переглядати, контролювати й управляти вузлами через єдиний інтерфейс із функціями пакетних команд і автоматичного оповіщення.

Таким чином, SNMP рятує адміністратора від необхідності введення команд вручну. Усього були розроблені й розгорнуто три версії. Усі вони використовуються дотепер, а найпоширенішої стала друга – SNMPv2c.

Архітектура SNMP

Компоненти, що становлять архітектуру SNMP:

- мережна станція керування, що включає в себе мережного менеджера;
- агенти;
- майстер-агенти;
- керовані компоненти.

Мережна станція керування – NMS

Network Management Station (NMS) віддалено моніторить керовані пристрої, одержує дані, зібрані майстер-агентами, відслідковує продуктивність і

представляє отриману інформацію в графічному виді. Вбудований менеджер NMS відповідає за зв'язок з агентами.

Агенти

Майстер-агент

Це програма, що зв'язує мережних менеджерів і субагентів. Майстер-агент аналізує запити мережного менеджера NMS і пересилає їх субагентам, збирає й формує відповіді субагентів і відправляє їхньому менеджерові. Майстер-агент повідомляє менеджера, якщо запит некоректний або запитана інформація недоступна.

Субагент

Це програма, що поставляється вендором разом з мережним пристроєм. Субагент пересилає зібрану інформацію майстрові-агентові. У кожного керованого компонента є відповідний субагент.

Керований компонент

Це підключений до мережі пристрій або програмне забезпечення із вбудованим субагентом. До таких пристроїв ставляться не тільки маршрутизатори, комутатори й сервери, але й IP-відеокамери, БФП й IP-телефони. До софту із субагентами також ставляться антивірусні програми, системи резервного копіювання, ПЗ для систем ІБЖ.

База керуючої інформації – MIB

MIB – це ієрархічна база даних з відомостями про пристрій. У кожного типу пристрою своя Mib-таблиця: у принтера в ній утримується інформація про стан картриджів, а в комутатора – дані про трафік. Завдяки MIB менеджер знає, яку інформацію він може запросити в агента пристрою.

Ідентифікатор об'єкта – OID

Кожний об'єкт в MIB має свій унікальний ID – OID, який представлений у числовому форматі й має ієрархічну структуру. OID – це числовий еквівалент шляху до файлу. Він привласнює значення кожній таблиці в MIB, кожному стовпцю в таблиці й кожному значенню в стовпці.

Наприклад, OID 1.3.6.1.4.868.2.4.1.1.1.3.3562.3. означає

iso.org.dod.internet.private.transition.products.chassis.card.slotcps.
cpsslotsummary.cpsmoduletable.cpsmoduleentry.cpsmodulemodel.3562.3.

Використовуючи перші 6 цифр цього OID, можна пройти по дереву на схемі.

Частина значень в OID містить дані про виробника пристрої, що дозволяє швидко одержати певну інформацію про девайс.

Деревоподібна ієрархія MIB і OID в SNMP виглядає трохи заплутаної, але в неї є свої переваги. Це проста й гнучка система організації мережних пристроїв, вона працює незалежно від розміру мережі.

Теорія й логіка роботи протоколу SNMP

Призначення

Споконвічно протокол повинен був надати системним адміністраторам інструмент для керування інтернетом. Однак, гнучка архітектура SNMP дозволила проводити моніторинг усіх мережних пристроїв і управляти ними з однієї консолі. Це й стало причиною поширення SNMP.

PDU

Менеджери й агенти обмінюються даними через протокол UDP. Замість нього також може використовуватися TCP, IPX або протокол MAC-Рівня. Обмін даними заснований на Protocol Data Unit (PDU).

Усього в SNMP сім PDU:

- GET – запит менеджера NMS на одержання даних с пристрою.
- GETNEXT – запит, аналогічний GET. Відмінність лише в тому, що менеджер запитує дані, що перебувають на наступному рівні ієрархії OID, в MIB.
- SET – за допомогою цього запиту менеджер змінює або привласнює пристрою нові дані.

– RESPONSE – повідомлення від агента, що висилається у відповідь на запит даних.

– TRAP – повідомлення про що відбувся події або помилці. Агент відправляє його відразу після настання події, не чекаючи запиту менеджера. Менеджер ніяк не підтверджує одержання повідомлення, що може стати проблемою.

– GETBULK – запит агентіві на добування із пристрою масиву даних. Це поліпшений варіант запиту GETNEXT.

– INFORM – повідомлення, аналогічне TRAP, але з підтвердженням одержання. Агент буде відправляти повідомлення, поки менеджер не підтвердить, що воно дійшло.

TRAP, GETBULK – є тільки в другий і третьої версіях протоколу SNMP.

Таблиця 3.1 – Схема PDU

IP заголовок	TCP/IP	TCP/IP
UDP заголовок	TCP/IP	TCP/IP
Версія SNMP	v1/v2/v3	PDU
Рядок співтовариства	Public, Private	PDU
Тип PDU	Get, Getnext, Response, Set, Trap, Getbulk, Inform	PDU
ID запиту	Ідентифікатор запиту	PDU
Статус помилки	0, 1, 2, 3, 4, 5	PDU
Індекс помилки	0, 1	PDU
Зв'язані змінні	Одна або трохи змінних у запиті	PDU

Застосування

Статуси помилок і їх опис.

– 0 – noerror – Процес завершений успішно.

– 1 – toobig – Об'єкт занадто великої й не міститься в повідомлення Response.

– 2 – nosuchname – Для запитів GET і SET: запитана змінна не існує в базі MIB. Для запитів GETNEXT: змінна не має приймача в дереві MIB.

– 3 – badvalue – Для запитів SET: зроблена помилка в синтаксисі або задане неприпустиме значення.

– 4 – readonly – Помилка не визначена.

– 5 – generr – Інші помилки, наприклад, спроба привласнити значення, що перевищує межі реалізації.

Мережні порти SNMP

За замовчуванням SNMP використовує Udp-порти 161 і 162. Менеджер відправляє запити на порт 161 агента. З порту 161 агент відправляє відповідь менеджерові. При відправленні запиту менеджер додає до нього ID, а агент вставляє цей ID у відповідь, щоб менеджер міг зв'язати свій запит з відповіддю агента.

Пастки агент висилає на порт 162 менеджера. Якщо використовується DLTS або TLS, то агент висилає повідомлення на порт 10162, а менеджер – на порт 10161. Адміністратор може змінити порти SNMP, використовувані за замовчуванням, на будь-які інші.

Пастки

Пастка (Trap) – це найважливіший спосіб комунікації в SNMP. Менеджер відповідає за велику кількість пристроїв, на багатьох з них може бути кілька керованих компонентів. Агент відправляє пастку зі своєї ініціативи, коли необхідно повідомити менеджера про подію. Наприклад, пастка може вислати звіт про перегрів машини або про те, що в тонері закінчилося чорнило.

Одержавши повідомлення, менеджер вибирає потрібну дію, наприклад, опитує агента, щоб одержати повну виставу про те, що відбулося. Перелік повідомлень, які посилає пастка:

– 0 – coldstart – Холодний запуск пристрою.

– 1 – warmstart – Гарячий запуск пристрою.

– 2 – linkdown – Інтерфейс відключився.

– 3 – linkup – Інтерфейс увімкнувся.

– 4 – authenticationfailure – Менеджер вислав повідомлення з невірним рядком співтовариства.

– 5 – egpneighborloss – Агент втратив зв'язок з хостом по протоколу Exterior Gateway Protocol (EGP).

– 6 – entrprisespecific – Відбулася подія, характерне для виробника даного пристрою.

В SNMP є два типи пасток: Trap і Inform. Відмінності між ними в тому, що після одержання Inform менеджер підтверджує одержання пастки. А якщо ні, то агент буде відправляти Inform, поки не одержить підтвердження. А от після одержання Trap менеджер не відправляє підтвердження. Якщо повідомлення не дійшло до менеджера, агент про це не довідається.

Версії протоколу SNMP

SNMPv1

Перша версія протоколу створена в 80-х роках XX століття. Легка в налаштовуванні – потрібно тільки рядок community. Версія широко використовується дотепер.

SNMPv2c

Друга версія протоколу SNMP з'явилася в 1993 році. Розроблювачі додали в неї новий запит Getbulk і пастку Inform, а також удосконалили безпеку.

У цієї версії є два способи комунікувати із пристроями, що підтримують SNMPv1: двомовна система мережного керування й проксі-агенти. Проксі-агенти виконують роль майстер-агентів, а у двомовній системі керування менеджер визначає, яку версію SNMP підтримує агент, і зв'язується з ним через SNMPv1 або SNMPv2c.

SNMPv3

Третя версія вийшла в 1998 році. Розроблювачі додали в SNMP криптографічний захист, полегшили віддалене налаштовування й адміністрування об'єктів. Цього вдалося досягтися за рахунок визначення набору

стандартизованих об'єктів керування, називаних об'єктами MIB віддаленого мережного моніторингу, – RMON MIB.

Безпека

Споконвічно безпека не була першочерговий завданням розроблювачів. Перша версія SNMP була створена для віддаленого адміністрування в часи, коли погроза несанкціонованого доступу була мінімальною. Тому SNMPv1 слабко захищена від злому, і зловмисники могли використовувати її для проникнення в мережну інфраструктуру.

У роботі над другою версією протоколу розроблювачі запропонували кілька варіантів рішення. Поширення одержав варіант SNMPv2c – не самий надійний, але простій у використанні.

Основна проблема з безпекою в тому, що майже все устаткування підтримує версію SNMPv1. І тільки частина працює з версіями SNMPv2c і SNMPv3. Саме тому самої популярної стала SNMPv2c. Вона здатна працювати із пристроями, які підтримують першу або другу версії SNMP.

Таблиця 3.2 – Моделі безпеки протоколів SNMP по версіях

SNMPv1	Community–based security
SNMPv2c	Community–based security
SNMPv2u	User–based security
SNMPv2	Party–based security
SNMPv3	User–based security

Community-based Security – модель безпеки на основі рядка співтовариства. Фактично це ідентифікатор користувача або пароль, який відправляється разом із запитом. Якщо рядок співтовариства невірний, агент ігнорує запит.

Рядок співтовариства залежить від вендора пристрою. Часто вендори за замовчуванням вибирають «PUBLIC» у якості пароля, тому першою справою на

нових пристроях потрібно змінити рядок співтовариства для захисту мережі від зловмисників.

Рядка співтовариства бувають трьох видів:

- тільки для читання – дозволяє одержувати дані із пристрою;
- читання/запис – дозволяє одержувати дані й змінювати конфігурацію пристрою;
- рядок співтовариства SNMP Trap – дозволяє одержувати пастки.

Рядок співтовариства широко використовується через свою простоту й наявності зовнішніх систем безпеки.

Party-based Security Model – модель на основі так званих «сторін». Для комунікації між менеджером і агентів вибирається логічна сутність, називана стороною. Вона встановлює протоколи автентифікації й шифрування, а відправник і одержувач погоджуються зі способом шифрування й дешифрування даних. Складність використання моделі перешкодила її поширенню серед користувачів.

User-based Security Model – модель безпеки на основі користувачів. Рівні автентифікації, безпеки й конфіденційності протоколів і ключів налаштовуються в агента й менеджера.

Найкраще безпека пророблена в третій версії SNMP за рахунок USM і VACM. Спрощено VACM (View-based Access Control Model) можна описати як модель із різними рівнями доступу для груп менеджерів. При одержанні запиту агент вирішує, чи дозволений певній групі менеджерів доступ до читання, запису й одержанню повідомлень.

Типові проблеми безпеки:

- Периметр мережі може бути небезпечний, якщо запити SNMP дозволені міжмережевими екранами й пакетними фільтрами.
- При активації функцій SNMP на деяких пристроях ім'я рядка співтовариства за замовчуванням PUBLIC. Хакер почне пошук саме із цього.

– Припинення відправлення пасток. Змінивши запис у команді SnmpEnableAuthenTraps, зловмисник може припинити відправлення пасток. У випадку невдалої автентифікації він може не турбуватися про те, що його безуспішні спроби злому привернуть увагу адміністратора мережі.

– Віддалене пакетне перехоплення за допомогою сніферів – програм аналізу мережного трафіку.

– Слабкий контроль доступу до рядка співтовариства читання-запис. Вона дає всім користувачам можливість змінювати конфігурацію пристроїв мережі SNMP. Адміністратор повинен уважно стежити за цим, інакше безконтрольну зміну конфігурацій допоможе зловмисникові завдати шкоди системі.

Якщо системний адміністратор не використовує SNMP, то він повинен відключити його на пристроях.

Практичне застосування протоколу

За допомогою SNMP адміністратор управляє застосункум і хмарними сервісами, адмініструє локальну мережу й контролює стан сервера з однієї консолі.

Можливості SNMP-протоколу

Завдяки протоколу адміністратор може:

– віддалено скидати паролі й переналаштовувати IP-адреси;

– збирати інформацію про навантаження на пропускну здатність мережі;

– відправляти запити для моніторингу мережних пристроїв;

– одержувати повідомлення про простір, що закінчується, на диску;

– відслідковувати навантаження на CPU сервера й одержувати повідомлення про перевищення припустимого порога;

– одержувати повідомлення про несправність підключеного до мережі пристрою;

– у реальному часі одержувати повідомлення про помилки на пристроях;

– збирати інформацію про помилки.

За допомогою стороннього ПЗ можна також:

- управляти хмарними сервісами;
- сканувати по діапазону IP-адрес;
- додавати дані через кастомні OID.

SNMP і перехід з IPv4 на IPv6

Протокол за замовчуванням повинен працювати з IPv4 або IPv6. На практиці IPv6 створює певні проблеми для роботи SNMP. Ці проблеми зв'язані об'єктами MIB, що містять мережні адреси. OID в MIB зберігають інформацію для декількох рівнів TCP/IP, і відмінності між IPv4 і IPv6 будуть відбиті в OID.

Відсутність підтримки IPv6 в існуючих об'єктах MIB проявляється двома способами:

- об'єкти MIB підтримують тільки IPv4, але не IPv6;
- що втримуються в OID IPv 4-адреси не обов'язково являють собою IP-адреса.

Ці проблеми вирішуються також двома способами:

- створенням нових баз MIB з підтримкою тільки IPv6 або незалежних від версій протоколу Protocol-version independent (PVI);
- модифікацією MIB для додавання нових або відновлення існуючих OID з підтримкою IPv6.

Інсталяція

Налаштовування SNMP в Windows

Налаштовування даних агента SNMP

Пуск → Панель керування → Адміністрування → Керування комп'ютером.

- У дереві консолі треба розгорнути вузол «Служби й застосунку» і вибрати пункт «Служби».
- В області праворуч двічі клацнути елемент «Служба SNMP».
- Потім відкрити вкладку «Агент».

– Увести ім'я користувача або адміністратора комп'ютера в поле «Контакт», а потім увести фізичне розташування комп'ютера або контакту в поле «Розташування». Ці коментарі обробляються як текст і є необов'язковими.

– У розділі «Служба» треба встановити прапорці поруч зі службами, надаваними комп'ютером і нажати «ОК».

Налаштовування співтовариства й пасток SNMP

Пуск → Панель керування → Адміністрування → Керування комп'ютером.

– У дереві консолі треба розгорнути вузол «Служби й застосунку» і вибрати пункт «Служби».

– В області праворуч двічі клацнути елемент «Служба SNMP».

– Відкрити вкладку «Трепінг».

– У полі «Ім'я співтовариства» увести ім'я співтовариства й нажати кнопку «Додати в список».

– У розділі «Адресати пасток» нажати кнопку «Додати».

– У полі «Host Name» увести ім'я, IP-адреса вузла й нажати «Додати». Ім'я вузла або адреса з'явиться в списку призначення пасток.

– Нажати «ОК».

Налаштовування безпеки SNMP

Пуск → Панель керування → Адміністрування → Керування комп'ютером.

– У дереві консолі потрібно розгорнути вузол «Служби й застосунку» і вибрати пункт «Служби».

– В області праворуч двічі клацнути елемент «Служба SNMP».

– Відкрити вкладку «Безпека».

– Установити прапорець «Пересилання пасток перевірки дійсності», якщо необхідно, щоб агент відправляв пастку при збої перевірки дійсності.

– У розділі «Прийнятні імена співтовариств» треба нажати кнопку «Додати».

– У полі «Права співтовариства» вибрати дозволи, щоб указати, як вузол буде обробляти запити SNMP від обраного співтовариства.

– У полі «Ім'я співтовариства» увести потрібне ім'я співтовариства з урахуванням регістру, а потім нажати кнопку «Додати».

– Потім, щоб ухвалювати запити SNMP від будь-якого вузла в мережі, незалежно від їхнього посвідчення, треба вибрати варіант «Ухвалювати пакети SNMP з будь-якого вузла».

– Щоб обмежити прийняття пакетів SNMP, потрібно нажати «Ухвалювати пакети SNMP із цих комп'ютерів», потім нажати «Додати» і ввести в поле ім'я вузла, IP-адреса або Ірх-адреса відповідного вузла. Нажати «Додати», а потім «ОК».

Налаштовування SNMP в Linux

Налаштовування SNMP в Centos 7

Спочатку потрібно встановити останні відновлення за допомогою `yum/dnf`:

```
yum update
```

потім установити SNMP:

```
yum install net-SNMP net-SNMP-utils
```

і створити копію конфігураційного файлу:

```
mv /etc/SNMP/SNMPd.conf /etc/SNMP/SNMPd.conf.orig
```

тепер потрібно відредагувати налаштовування агента

```
nano /etc/SNMP/SNMPd.conf
```

і додати рядка:

```
community public
syslocation Mylocation
syscontact admin@example.com
```

Локацію й email краще вказати реальні.

Настав час додати сервіс в автозавантаження й запустити знову його:

```
systemctl enable SNMPd.service
systemctl start SNMPd
```

Як перевірити, що сервіс запущений:

```
systemctl status SNMPd
```

Опитування агента за допомогою утиліти SNMPwalk:

```
SNMPwalk -v 2c -c public -O e 127.0.0.1
```

Опитування сервера локально командою:

```
SNMPwalk -v2c -c public localhost system
```

Налаштовування SNMP в Debian 10

Спочатку потрібно встановити демона, клієнта й файли:

```
apt install SNMPd SNMP libSNMP-dev
```

Після установки переходимо до налаштовування SNMP в Debian.

Файлом налаштовування SNMP-агента за замовчуванням є `/etc/SNMP/SNMPd.conf`. Агент SNMP може бути запущений з налаштовуваннями за замовчуванням. Однак для включення віддаленого моніторингу потрібно зробити кілька змін. Для цього створіть резервну копію файлу:

```
cp /etc/SNMP/SNMPd.conf /etc/SNMP/SNMPd.conf.orig
```

Тепер потрібно змінити директиву agentadress. Її поточні налаштовування дозволяють доступ тільки з локального комп'ютера. Для включення віддаленого моніторингу необхідно визначити IP-адресу інтерфейсу:

```
vim /etc/SNMP/SNMPd.conf
###########################################################################
#
#  AGENT BEHAVIOUR
#


#  Listen for connections from the local system only
agentaddress  udp:127.0.0.1:161,udp:192.168.43.62:161
```

Для налаштовування автентифікації:

```
directive community ]
```

rocommunity надає доступ тільки на читання, а rwcommunity дає доступ до читання/запису. В Access Control section потрібно помістити рядок

```
rocommunity S3Cure 192.168.43.100
```

Крім того, можна включити запит з локального хосту rocommunity S3Cure localhost:

```
rouser   authonlyuser
rwuser   authprivuser   priv
rocommunity S3Cure localhost
rocommunity S3Cure 192.168.43.100
```

Потім потрібно запустити знову SNMP:

```
systemctl restart SNMPd
```

Щоб додати сервіс в автозавантаження, уведіть:

```
systemctl enable SNMPd
```

SNMP – це простий і ефективний спосіб для збору й обміну інформацією між мережними пристроями, які випущені різними вендорами й працюють на різному ПЗ. Цей протокол – не ідеальне, але усе ще одне із кращих застосунків для моніторингу й керування. На сьогоднішній день немає іншого інструмента з порівнянним рівнем підтримки й використання.

Створений 30 років тому SNMP продовжує працювати, тому що він має характеристики, яких немає ні в однієї з його аналогів. Він простій у використанні, безкоштовний і підтримується практично всіма вендорами.

3.2 Розробка структурної схеми

Network Performance Monitor (NPM) – потужний, але простий у використанні застосунку для керування доступністю й продуктивністю мережі, яке надає операторові всі необхідні ключові відомості про мережу. NPM допомагає швидко виявляти, діагностувати й усувати проблеми продуктивності й

збої в мережах. Крім того, система NPM характеризується винятковою простотою розгортання, експлуатації й обслуговування. Це означає, що ви можете приділити більше часу безпосередньому керуванню мережею, а не обслуговуванню інструментів керування мережею. Крім того, самостійне розгортання, що займає всього одна година, дозволяє відразу приступитися до моніторингу, а не наймати додаткових консультантів.

Основні характеристики NPM:

– Комплексний моніторинг доступності й продуктивності мережі з можливістю складання звітів про мережу з устаткуванням різних виробників – моніторинг, відстеження стану працездатності, аналіз у режимі реального часу, докладний аналіз, статистика продуктивності мережі по маршрутизаторах, комутаторах, крапках бездротового доступу, серверам, віртуальним середовищам і пристроям, що використовують протокол SNMP, в інтуїтивно зрозумілому користувацькому веб-інтерфейсі, що надає детальні дані про кожний пристрій і докладні відомості про систему.

– Автоматичне виявлення мережних пристроїв і зіставлення – автоматичне виявлення змін у мережі з миттєвим включенням нових пристроїв у моніторинг. Графічне відображення структури мережі з візуальним відстеженням стану пристроїв.

– Інтелектуальне оповіщення про події мережі – одержуйте попередження про реальні неполадки з використанням складного алгоритму аналізу на основі кореляції подій і попередньо заданих умов.

– Гнучке керування станом служб – динамічне групування мережних ресурсів з метою моделювання ключових служб і комплексне керування службами поряд з інфраструктурою мережі.

Ключові переваги:

1. Прискорення пошуку й усунення несправностей, підвищення рівня обслуговування й скорочення часу простою

2. Моніторинг і відображення часу відгуку, доступності й продуктивності мережних пристроїв

3. Розширена візуальна вистава мережі з автоматичним плануванням ресурсів, картами продуктивності бездротових мереж і інтелектуальними оповіщеннями із прив'язкою до топології

4. Підвищення експлуатаційної ефективності завдяки вбудованим панелям моніторингу, оповіщенням і звітам

5. Автоматичне виявлення мережних пристроїв і стандартне розгортання менш чому за годину

Моніторинг доступності й продуктивності мережі

Відслідковує доступність мережних пристроїв і інтерфейсу, а також показники продуктивності, такі як використання смуги пропущення, втрата пакетів, затримки, помилки, відкидання, ЦП і пам'ять для пристроїв, що підтримують SNMP і WMI.

Поглиблена перевірка пакетів і якість сприйняття

Видає візуальне зведення метрик продуктивності мережі й застосунків шляхом поглибленої перевірки й аналізу пакетів. Швидко виявляє зниження продуктивності застосунків або зміни в їхній роботі, а також визначає, що стало причиною змін – застосунок або мережа.

Автоматичне виявлення мережних пристроїв

Становить розклад сканування мережі за допомогою простого у використанні майстра виявлення з веб-інтерфейсом. Виявляє нові мережні пристрої й забезпечує можливість моніторингу стану всього важливого устаткування.

Динамічна топологія, що й налаштовується, мережі

Дає можливість налаштовування топології мережі шляхом перетаскування мережних пристроїв і автоматичного перегляду з'єднань, використання каналів підключення між пристроями на 2-м рівні і їх стану в реальному часі.

Планування ресурсів мережі

Можна легко управляти мережею й оптимізувати її інфраструктуру, одержуючи оповіщення про можливе вичерпання ключових мережних ресурсів, наприклад, смуги пропущення, пам'яті, дискового простору й т.п.

Підтримка пристроїв різних виробників

Вбудовані засоби підтримки пристроїв різних виробників, установлюваних у сучасних складні мережних середовищах, що й бурхливо розвиваються.

Моніторинг працездатності устаткування

Одержуйте оперативну інформацію про стан мережного устаткування завдяки моніторингу, оповіщенням і звітам про стан ключових датчиків пристроїв, зокрема таких показників, як температура, швидкість обертання вентилятора й електроживлення.

Панель моніторингу стану й продуктивності застосунків

Переглядайте графіки й таблиці з 10 ключовими параметрами, а також докладні відомості про застосунки, список вузлів з застосункуми, які їх використовують, і зведення про вузли, на яких перевищені встановлені граничні значення.

Датчики для поглибленої перевірки й аналізу пакетів

Швидке й просте розгортання датчиків перевірки й аналізу пакетів за допомогою покрокового майстра з метою моніторингу продуктивності застосунків і мережі.

Інтелектуальне оповіщення про події в мережі

Швидке налаштовування повідомлень про взаємозалежні події умовах, що встановилися, і складних комбінаціях станів пристроїв.

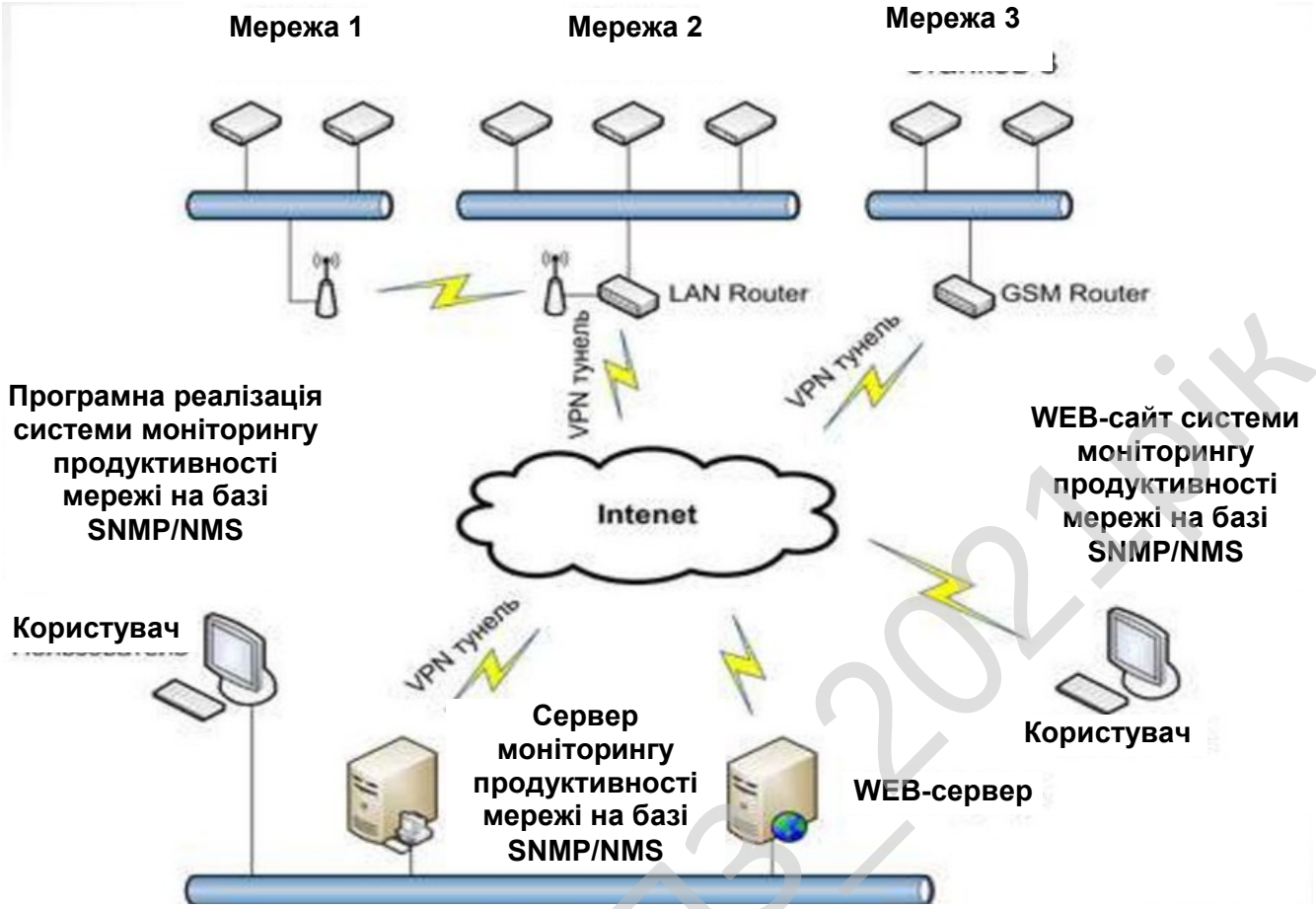

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

На відміну від великого різноманіття застосунків для керування й моніторингу IT-інфраструктури на основі SNMP і глобальних систем керування мережною інфраструктурою (NMS), застосунки для моніторингу й аналізу продуктивності мережі мають ряд ключових відмінностей і мають наступні функціональні можливості:

– Пасивний метод моніторингу при якому аналізується реальний трафік користувачів у мережі за допомогою апаратних пробників або програмно-апаратних комплексів на базі сервера зі спеціалізованими модулями захвата й аналізу трафіку. Дані пристрої підключаються до SPAN портам комутаторів або

через спеціалізовані розгалуджувачі або брокери пакетів у будь-які канали зв'язку.

– Збір і аналіз інформації про використання каналів зв’язку на основі Flow технологій, які присутні в комутаторах 3 рівня або маршрутизаторах.

– Можливість переглядати вміст мережних пакетів, і одержувати достатню інформацію для аналізу проблем на всіх рівнях моделі OSI, особливо в контексті VoIP, відео й багаторівневих бізнес застосунків. Це означає, що застосунок повинний мати можливість декодувати й представляти дані про запити користувачів до баз даних або Web порталам, а не просто зберігати неопрацьовані пакетні дані для наступного аналізу за допомогою сторонніх додаткових застосунків.

– Можливість створювати, виконувати й одержувати примусові тестові запити (синтетичні запити), які використовують вбудовані в устаткування й операційні системи агенти або технології, наприклад: Service Assurance Agent (SSA), IP SLAs і Medianet.

– Здатність генерувати панелі керування, які зручні для ознайомлення керівниками компаній, що й інформують про поточний стан ключових сервісів і стані IT-інфраструктури.

– Доступність, що налаштовуються користувачем, передналаштованих або інтелектуальних граничних значень, при перевищенні яких система почне процес розслідування або повідомить відповідальних осіб про виникнення події.

– Кореляція всіх подій за часом, які допоможуть оцінити стан елементів інфраструктури в момент виникнення інциденту.

– IT потрібно більш глибоке розуміння й контроль над продуктивністю мережі і її використанням з погляду ключових сервісів для ефективного планування витрат і обміркованого ухвалення рішення для витрат на оптимізацію й модернізацію IT-інфраструктури.

– Застосунки для діагностики й моніторингу продуктивності мережі з погляду надаваних сервісів надає IT-фахівцям і IT-директорам можливість

виявлення недовантажених або перевантажених мережних ресурсів, які можуть бути використані або своєчасне модернізовані для підтримки нових сервісів.

– Підвищень інтерес до застосунків для віртуалізації, хмар, уніфікованих комунікацій накладають додаткові вимоги до застосунків NPM.

– Програмні застосунки для глибокого аналізу пакетів накладають відбиток на вартості існуючих застосунків для діагностики й аналізу продуктивності IT-інфраструктури.

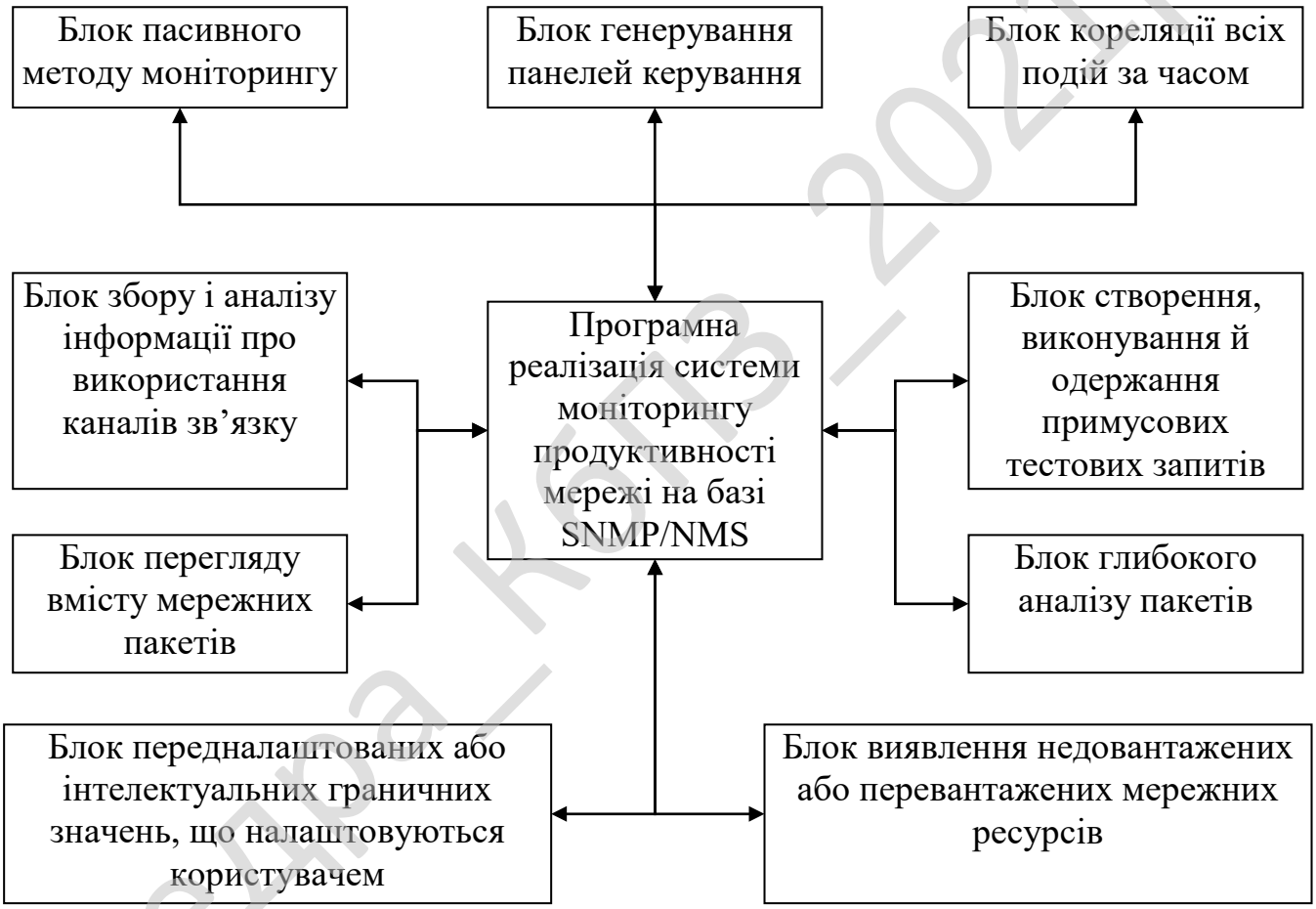

Рисунок 3.2 – Функціональна схема системи

Рекомендації до дії:

– Замовити пілотний проект і оцінити можливості NPM застосунків у частині вбудованої аналітики й зручності деталізації для пошуку дійсних причин

зниження продуктивності, щоб дати можливість IT-фахівцям бути більш проактивними.

– Інвестувати в застосунки NPM інструменти з метою застосунки основного завдання бізнесу для поліпшення продуктивності якості сервісу й часу відгуку застосунків.

– Використовувати економічно ефективний баланс між апаратними розв'язками для моніторингу сервісів у власному ЦОД і програмними розв'язками для аналізу завантаження каналів зв’язку з віддаленими офісами й моніторингу продуктивності хмарних застосунків IaaS, PaaS, SaaS.

– Використовувати NPM застосунки для планування й керування ємністю каналів зв’язку при впровадженні вимогливих до пропускної здатності й затримкам застосунків, таких як передача голосу по IP і уніфіковані комунікації.

Що необхідно знати при виборі застосунків для моніторингу продуктивності мережі?

На відміну від використовуваних застосунків для керування мережею (SNMP pooling), застосунки для моніторингу продуктивності мережі дозволяють IT розуміти поведінка всієї IT-інфраструктури як середовища для надання ключових бізнес сервісів, а не ухвалювати застосунки в реактивному режимі по розширенню каналів зв’язку й збільшенню продуктивності окремих елементів мережі в авральному режимі.

Оцінка й аналіз продуктивності мережі мають важливе значення як для бізнесу, так і для IT і впливає на задоволеність і ефективність користувачів. І нарешті, застосунки NPM дозволяють поліпшувати ємність мережі шляхом прийняття зважених застосунків і, тим самим, знижуючи капітальні витрати в мережне устаткування.

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3.

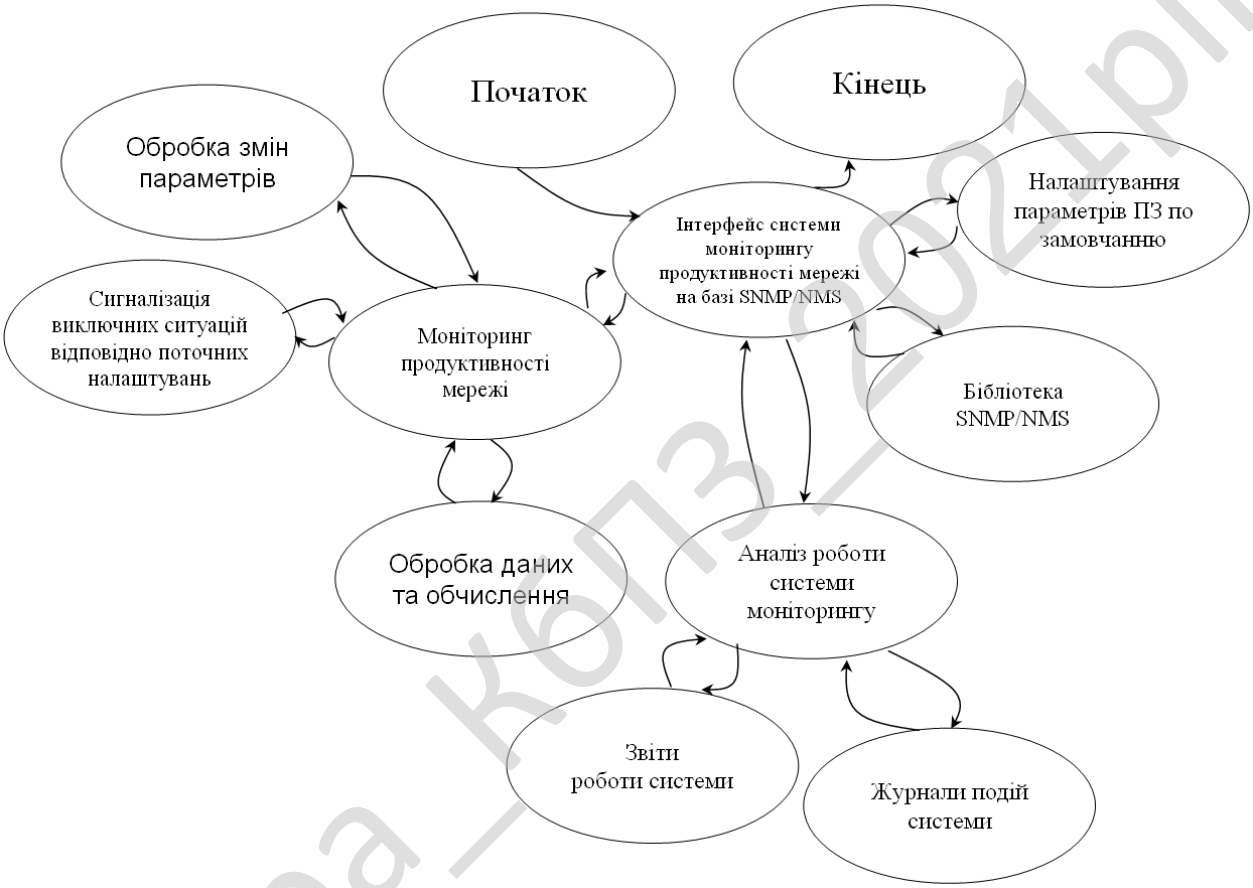

Рисунок 3.3 – Діаграма взаємодії процесів

Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, поток сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду 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

Під час роботи над програмою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних - прямокутником, блок прийняття рішень - ромбом, еліпсом - початок та кінець алгоритму.

Розглянемо алгоритм роботи основної програми у вигляді блок-схеми зображеної на рисунку 4.1. Вона складається з наступних функціональних блоків:

– Виведення вікна системи моніторингу продуктивності мережі.

– Сканування хостів мережі на базі SNMP/NMS (залежно від налаштувань).

– Додавання у БД нових хостів.

– Запит активних хостів.

– Виведення поточних хостів.

– Запит обробки даних хоста.

– Налаштування зв’язку з хостом мережі.

– Встановлення команд хоста на базі SNMP/NMS (залежно від налаштувань).

– Формування команд.

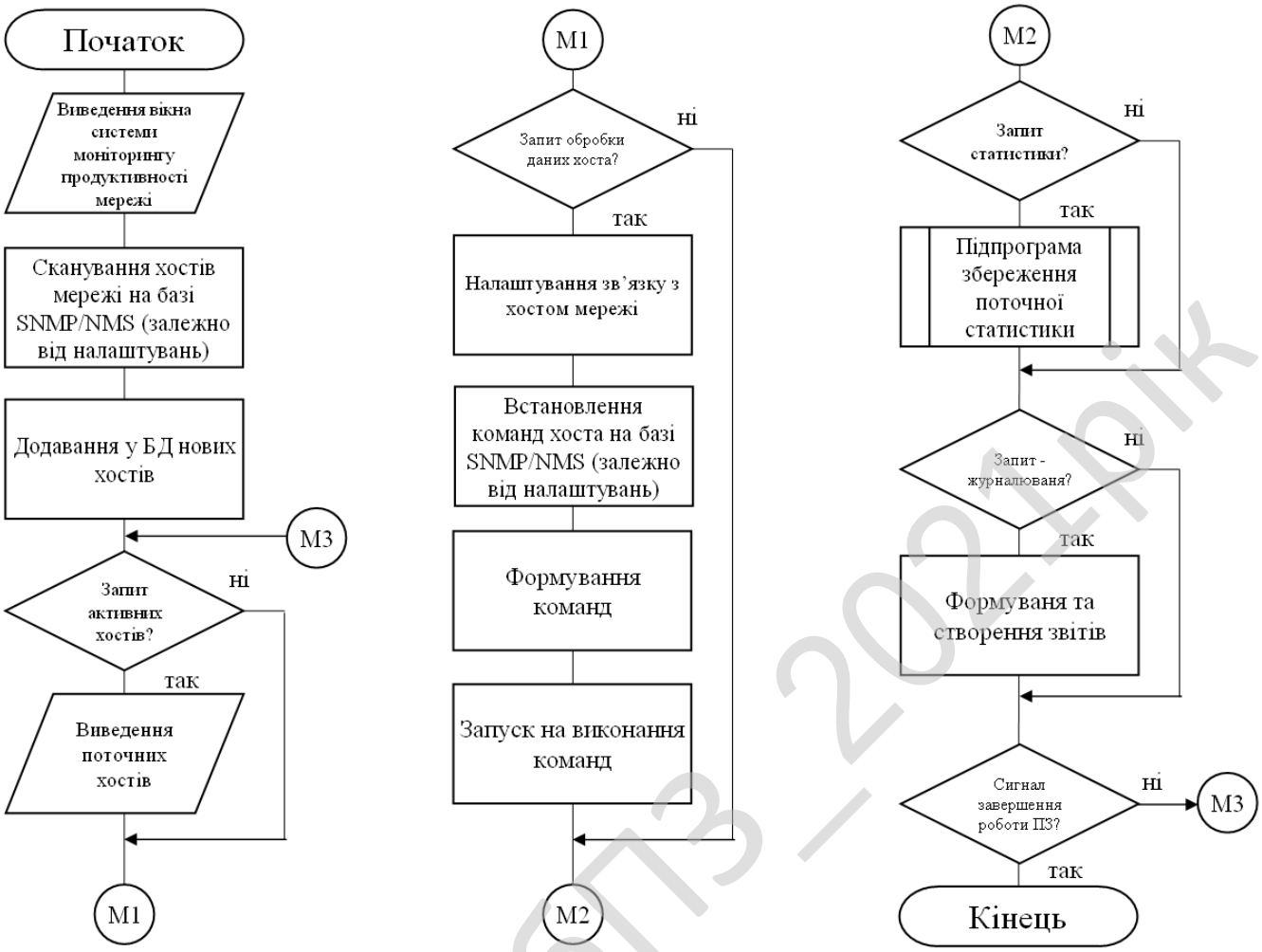

Рисунок 4.1 – Блок-схема основної програми

– Запуск на виконання команд.

– Запит статистики.

– Підпрограма збереження поточної статистики.

– Запит – журналюваня.

– Формування та створення звітів.

– Сигнал завершення роботи ПЗ.

На рисунку 4.2 зображено роботу підпрограми з наступними функціональними блоками:

– Виведення вікна системи моніторингу продуктивності мережі.

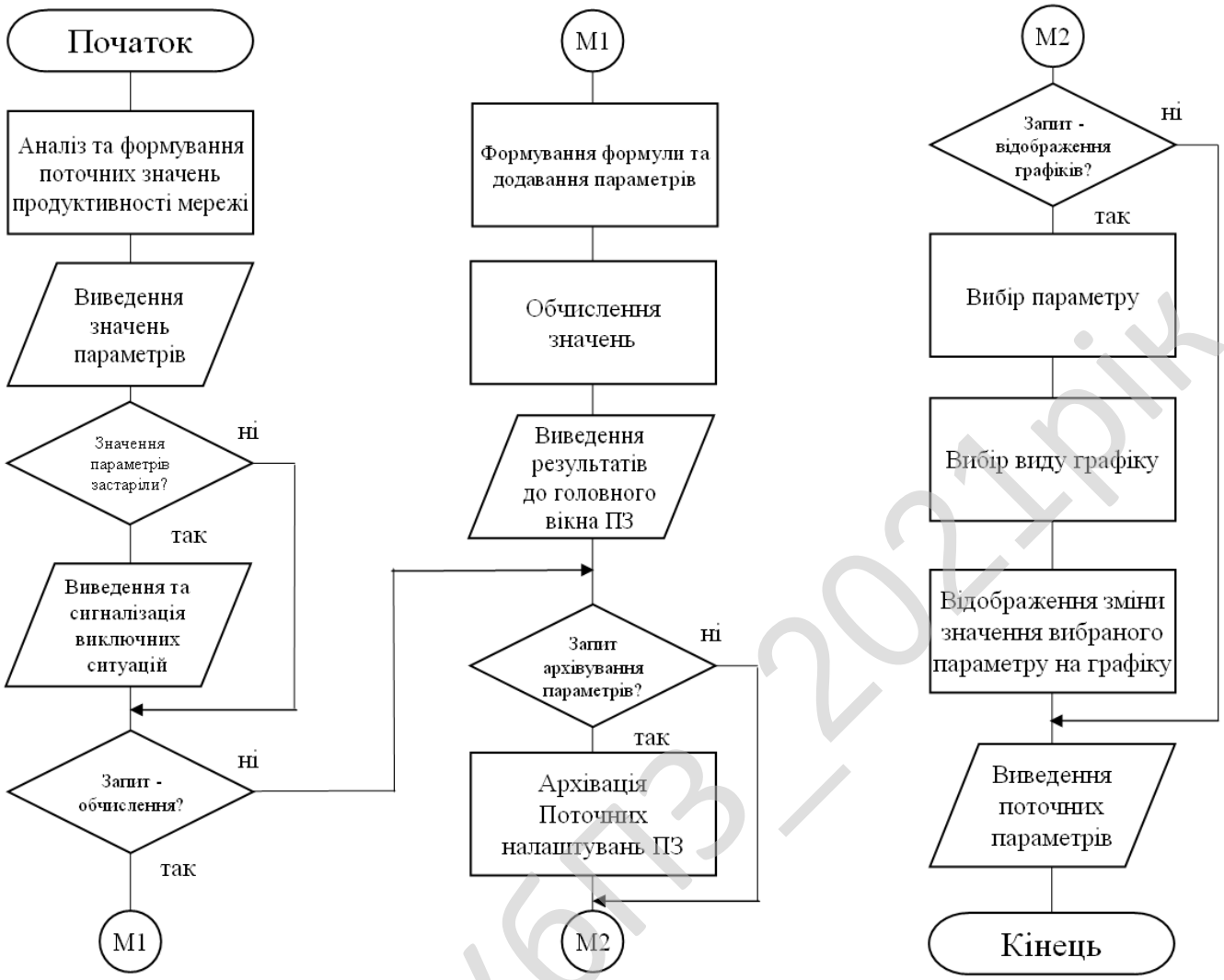

Рисунок 4.2 – Блок-схема роботи підпрограми

– Сканування хостів мережі на базі SNMP/NMS (залежно від налаштувань).

– Додавання у БД нових хостів.

– Запит активних хостів.

– Виведення поточних хостів.

– Запит обробки даних хоста.

– Налаштування зв’язку з хостом мережі.

– Встановлення команд хоста на базі SNMP/NMS (залежно від налаштувань).

– Формування команд.

– Запуск на виконання команд.

– Запит статистики.

– Підпрограма збереження поточної статистики.

– Запит – журналюваня.

– Формування та створення звітів.

– Сигнал завершення роботи ПЗ.

Розглянемо розроблений модуль системи моніторингу продуктивності мережі на базі SNMP/NMS, а саме модуль «uin» який призначено для отримання інформації про встановлені мережеві інтерфейси. Вихідний код модуля наступний:

```
unit uin; // назва uin

interface
/ описова частина
uses
  Windows, SysUtils, Classes, Controls, Forms, ComCtrls;

Const
// константи
  MAX_ADAPTER_NAME_LENGTH        = 256;
  MAX_ADAPTER_DESCRIPTION_LENGTH = 128;
  MAX_ADAPTER_ADDRESS_LENGTH     = 8;
  IPHelper = 'iphlpapi.dll';

  // типи адаптерів
  MIB_IF_TYPE_OTHER     = 1;
  MIB_IF_TYPE_ETHERNET  = 6;
  MIB_IF_TYPE_TOKENRING = 9;
  MIB_IF_TYPE_FDDI      = 15;
  MIB_IF_TYPE_PPP       = 23;
  MIB_IF_TYPE_LOOPBACK  = 24;
  MIB_IF_TYPE_SLIP      = 28;

type
  // Структури для виконання GetAdaptersInfo
  time_t = Longint;

  IP_ADDRESS_STRING = record
```

```
    S: array [0..15] of Char;
  end;
  IP_MASK_STRING = IP_ADDRESS_STRING;
  PIP_MASK_STRING = ^IP_MASK_STRING;

  PIP_ADDR_STRING = ^IP_ADDR_STRING;
  IP_ADDR_STRING = record
    Next: PIP_ADDR_STRING;
    IpAddress: IP_ADDRESS_STRING;
    IpMask: IP_MASK_STRING;
    Context: DWORD;
  end;

  PIP_ADAPTER_INFO = ^IP_ADAPTER_INFO;
  IP_ADAPTER_INFO = record
    Next: PIP_ADAPTER_INFO;
    ComboIndex: DWORD;
    AdapterName: array [0..MAX_ADAPTER_NAME_LENGTH + 3] of Char;
    Description: array [0..MAX_ADAPTER_DESCRIPTION_LENGTH + 3] of Char;
    AddressLength: UINT;
    Address: array [0..MAX_ADAPTER_ADDRESS_LENGTH - 1] of BYTE;
    Index: DWORD;
    Type_: UINT;
    DhcpEnabled: UINT;
    CurrentIpAddress: PIP_ADDR_STRING;
    IpAddressList: IP_ADDR_STRING;
    GatewayList: IP_ADDR_STRING;
    DhcpServer: IP_ADDR_STRING;
    HaveWins: BOOL;
    PrimaryWinsServer: IP_ADDR_STRING;
    SecondaryWinsServer: IP_ADDR_STRING;
    LeaseObtained: time_t;
    LeaseExpires: time_t;
  end;

  TfrmEnumNetInterfaces = class(TForm)
    tvInterfaces: TTreeView;
    procedure FormCreate(Sender: TObject);
  private
    procedure ReadLanInterfaces;
  end;
```

```
// За допомогою даної функції ми визначимо наявність мережевих інтерфейсів
// на локальному комп'ютері і інформацію про них
  function GetAdaptersInfo(pAdapterInfo: PIP_ADAPTER_INFO;
    var pOutBufLen: ULONG): DWORD; stdcall; external IPHelper;

var
  frmEnumNetInterfaces: TfrmEnumNetInterfaces;

implementation

{$R *.dfm}

// Читаємо всі IP адреси з усіх присутніх
// в системі мережевих інтерфейсів
procedure TfrmEnumNetInterfaces.ReadLanInterfaces;
// допоміжна функція перетворення
  function MACToStr(Addr: array of Byte; Len: Integer): String;
  var
    I: Integer;
  begin
    if Len = 0 then Result := '00-00-00-00-00-00' else
    begin
      Result := '';
      for I := 0 to Len - 2 do
        Result := Result + IntToHex(Addr[I], 2) + '-';
      Result := Result + IntToHex(Addr[Len - 1], 2);
    end;
  end;
// допоміжна функція перетворення
  function TimeToDateTimeStr(Value: Integer): String;
  const
    UnixDateDelta = 25569;
// приклад кількість днів між 12.31.1899 та 1.1.1970
    MinPerDay = 24 * 60;
    SecPerDay = 24 * 60 * 60;
  var
    Data: TDateTime;
    TimeZoneInformation: TTimeZoneInformation;
    AResult: DWORD;
  begin
    Result := '';
    if Value = 0 then Exit;
```

```
// Формат Unix TIME_T кількість секунд від 1.1.1970
    Data := UnixDateDelta + (Value / SecPerDay);
    AResult := GetTimeZoneInformation(TimeZoneInformation);
    case AResult of
      TIME_ZONE_ID_INVALID: RaiseLastOSError;
      TIME_ZONE_ID_STANDARD:
      begin
        Data := Data - ((TimeZoneInformation.Bias +
          TimeZoneInformation.StandardBias) / MinPerDay);
        Result := DateTimeToStr(Data) + ' ' +
          WideCharToString(TimeZoneInformation.StandardName);
      end;
    else
      Data := Data - ((TimeZoneInformation.Bias +
        TimeZoneInformation.DaylightBias) / MinPerDay);
      Result := DateTimeToStr(Data) + ' ' +
        WideCharToString(TimeZoneInformation.DaylightName);
    end;
  end;

var
  InterfaceInfo,
  TmpPointer: PIP_ADAPTER_INFO;
  IP: PIP_ADDR_STRING;
  Len: ULONG;
  AdapterTree, IPAddrTree, DHCPTree, WinsTree: TTreeNode;
  AdapterType: String;
begin
// Дивимося скільки пам'яті нам потрібно
  if GetAdaptersInfo(nil, Len) = ERROR_BUFFER_OVERFLOW then
  begin
// Беремо потрібну кількість
    GetMem(InterfaceInfo, Len);
    try
// виконання функції
      if GetAdaptersInfo (InterfaceInfo, Len) = ERROR_SUCCESS then
      begin
// Перераховуємо всі мережеві інтерфейси
        TmpPointer: = InterfaceInfo;
        repeat
// Ім'я мережевого інтерфейсу
```

```
AdapterTree: = tvInterfaces.Items.Add (nil, 'Adapted:' +
                TmpPointer ^ .AdapterName);
// Опис мережевого інтерфейсу
          tvInterfaces.Items.AddChild (AdapterTree, 'Description:' +
          TmpPointer ^ .Description);
// MAC Адреса
          tvInterfaces.Items.AddChild (AdapterTree, 'MAC:' +
            MACToStr (TmpPointer ^ .Address, TmpPointer ^ .AddressLength));
// Індекс адаптера в списку
          tvInterfaces.Items.AddChild (AdapterTree, 'Index:' +
            IntToStr (TmpPointer ^ .Index));
// Тип адаптера
          case TmpPointer ^ .Type_ of
            MIB_IF_TYPE_OTHER: AdapterType: = 'MIB_IF_TYPE_OTHER';
            MIB_IF_TYPE_ETHERNET: AdapterType: = 'MIB_IF_TYPE_ETHERNET';
            MIB_IF_TYPE_TOKENRING: AdapterType: = 'MIB_IF_TYPE_TOKENRING';
            MIB_IF_TYPE_FDDI: AdapterType: = 'MIB_IF_TYPE_FDDI';
            MIB_IF_TYPE_PPP: AdapterType: = 'MIB_IF_TYPE_PPP';
            MIB_IF_TYPE_LOOPBACK: AdapterType: = 'MIB_IF_TYPE_LOOPBACK';
            MIB_IF_TYPE_SLIP: AdapterType: = 'MIB_IF_TYPE_SLIP';
          else
            AdapterType: = 'Unknown';
          end;
          tvInterfaces.Items.AddChild (AdapterTree, 'Type:' + AdapterType);
// визначення активності DHCP
          if Boolean (TmpPointer ^ .DhcpEnabled) then
          begin
           DHCPTree: = tvInterfaces.Items.AddChild (AdapterTree,
                       'DHCP: Enabled');
// Адреса DHCP сервера
            tvInterfaces.Items.AddChild (DHCPTree, 'DHCP IP Addr:' +
              String (TmpPointer ^ .DhcpServer.IpAddress.S));
// Час отримання даних від сервера
            tvInterfaces.Items.AddChild (DHCPTree, 'LeaseObtained:' +
              TimeToDateTimeStr (TmpPointer ^ .LeaseObtained));
// Час старіння даних від сервера
            tvInterfaces.Items.AddChild (DHCPTree, 'LeaseExpires:' +
              TimeToDateTimeStr (TmpPointer ^ .LeaseExpires));
          end
          else
            tvInterfaces.Items.AddChild (AdapterTree, 'DHCP: Disabled');
// перераховуємо всі IP адреси інтерфейсу
```

```
          IP: = @ TmpPointer.IpAddressList;
          IPAddrTree: = tvInterfaces.Items.AddChild (AdapterTree, 'IP
                        Addreses:');
          repeat
            tvInterfaces.Items.AddChild (IPAddrTree, Format ( 'IP:% s,
            SubNetMask:% s', [String (IP ^ .IpAddress.S),
            String (IP ^ .IpMask.S)]));
            IP: = IP.Next;
          until IP = nil;
// Основний шлюз:
          tvInterfaces.Items.AddChild (AdapterTree, 'Default getaway:' +
            TmpPointer ^ .GatewayList.IpAddress.S);
// Windows Internet Name Service
          if TmpPointer ^ .HaveWins then
          begin
            WinsTree: = tvInterfaces.Items.AddChild (AdapterTree,
            'WINS: Enabled');
// основний WINS
            tvInterfaces.Items.AddChild (WinsTree, 'PrimaryWinsServer:' +
              String (TmpPointer ^ .PrimaryWinsServer.IpAddress.S));
// запасний WINS
            tvInterfaces.Items.AddChild (WinsTree, 'SecondaryWinsServer:' +
              String (TmpPointer ^ .SecondaryWinsServer.IpAddress.S));
          end
          else
            tvInterfaces.Items.AddChild (AdapterTree, 'WINS: Disabled');
          TmpPointer: = TmpPointer.Next;
        until TmpPointer = nil;
      end;
    finally
// Звільняємо зайняту пам'ять
      FreeMem (InterfaceInfo);
    end;
  end;
end;
// створення
procedure TfrmEnumNetInterfaces.FormCreate (Sender: TObject);
begin
  ReadLanInterfaces;
end;

end.
```

Розглянемо NetBIOS (Network Basic Input/Output System) – протокол для роботи в локальних мережах на персональних ЕОМ типу IBM/PC, розроблений у вигляді інтерфейсу, який не залежить від фірми–виробника. Був розроблений фірмою Sytek Corporation за замовленням IBM в 1983 році. Він включає в себе інтерфейс сеансового рівня (англ. NetBIOS interface), в якості транспортних протоколів використовує TCP і UDP.

Особливістю NetBIOS є можливість його роботи поверх різних протоколів, найпоширенішими/відомими з яких є NetBEUI, IPX і стек протоколів TCP/IP; причому якщо старі версії Windows орієнтувалися на більш легкі в реалізації і менш ресурсомісткі NetBEUI і IPX, то сучасні Windows орієнтуються на TCP/IP.

При використанні NetBEUI і IPX NetBIOS сам забезпечує надійність доставки даних (функціональність SPX не використовувати), а при використанні TCP/IP надійність доставки забезпечує TCP, за що удостоївся окремого імені «NBT».

Інтерфейс NetBIOS являє собою типовий інтерфейс взаємодії програм (API) для забезпечення мережевих операцій введення–виведення і управління транспортним протоколом.

Програми, що використовують NetBIOS API інтерфейс, можуть працювати тільки при наявності протоколу, що допускає використання такого інтерфейсу.

NetBIOS також визначає протокол, що функціонує на сеансовому/транспортному рівнях моделі OSI. Цей протокол використовується протоколами нижчих рівнів, такими як NBFP (NetBEUI) і NetBT для виконання мережевих запитів вводу–виводу і операцій, описаних в стандартному интерфейсном наборі команд NetBIOS.

Тобто, NetBIOS сам не підтримує виконання файлових операцій. Ця функція покладається на протоколи нижчих рівнів, а сам NetBIOS забезпечує тільки зв'язок з цими протоколами і NetBIOS API–інтерфейс.

Пакет NETBIOS створений для використання групою ЕОМ. Підтримує як режим сесій (робота через з'єднання), так і режим дейтаграм (без встановлення з'єднання). 16–символьні імена об'єктів в NETBIOS розподіляються динамічно.

NETBIOS має власну DNS, яка може взаємодіяти з інтернетівський. Ім'я об'єкта при роботі з NETBIOS не може починатися з символу *.

Додатки можуть знайти через NETBIOS потрібні їм ресурси, встановити зв'язок і послати або отримати інформацію. NETBIOS використовує для служби імен порт 137, для служби дейтаграм – порт 138, а для сесій – порт 139. Будь–яка сесія починається з NETBIOS–запиту, завдання IP–адреси і визначення TCP–порту віддаленого об'єкта, далі йде обмін NETBIOS–повідомленнями, після чого сесія закривається.

Сесія здійснює обмін інформацією між двома NETBIOS – додатками. Довжина повідомлення лежить в межах від 0 до 131 071 байт. Припустимо одночасне встановлення декількох сесій між двома об'єктами.

При організації IP–транспорту через NETBIOS IP–дейтаграмма вкладається в NETBIOS–пакет. Інформаційний обмін відбувається в цьому випадку без встановлення зв'язку між об'єктами. Імена NETBIOS повинні містити в собі IP–адреси. Так, частина NETBIOS–адреси може мати вигляд IP.XX.XX. XX. XX, де IP вказує на тип операції (IP через Netbios), а XX. XX. XX.XX – IP– адреса.

Система NETBIOS має власну систему команд (call, listen, hang up, send, receive, session status, reset, cancel, adapter status, unlink, remote program load) і примітивів для роботи з дейтаграммами (send datagram, send broadcast datagram, receive datagram , receive broadcast datagram).

Всі кінцеві вузли NETBIOS діляться на три типи:

– широкомовні («b») вузли;

– вузли точка–точка («p»);

– вузли змішаного типу («m»).

IP–адреса може асоціюватися з одним із зазначених типів. B–вузли встановлюють зв'язок зі своїм партнером за допомогою широкомовних запитів. P–

і M–вузли для цієї мети використовують netbios сервер імен (NBNS) і сервер розподілу дейтаграм (NBDD).

NetBIOS забезпечує:

– реєстрацію і перевірку мережевих імен;

– встановлення і розрив з'єднань;

– зв'язок з підтвердженням доставки інформації;

– зв'язок без підтвердження доставки інформації;

– підтримку управління і моніторингу драйвера і мережевої карти.

Розглянемо протокол SNMP (Simple Network Management Protocol, простий протокол керування мережею) – це протокол керування мережами зв'язку на основі архітектури TCP/IP.

На основі концепції TMN в 1980–1990 р. різними органами стандартизації був вироблений ряд протоколів керування мережами передачі даних з різним спектром реалізації функцій TMN. До одного з типів таких протоколів керування належить Simple Network Management Protocol (SNMP).

SNMP – це технологія, покликана забезпечити керування й контроль за пристроями й програмами в мережі зв'язку шляхом обміну керуючою інформацією між агентами, що розташовуються на мережних пристроях, і менеджерами, розташованими на станціях керування. SNMP визначає мережу як сукупність мережних керуючих станцій й елементів мережі (головні машини, шлюзи й маршрутизатори, термінальні сервери), які спільно забезпечують адміністративні зв'язки між мережними керуючими станціями й мережними агентами. SNMP різних версій присвячений цілий ряд рекомендацій IETF (RFC).

Зазвичай при використанні SNMP присутні керовані та керуючі системи. До складу керованої системи входить компонент, який називається агентом, який відправляє звіти керуючій системі. По суті SNMP агенти передають управлінську інформацію на керуючі системи як змінні (такі як «вільна пам'ять», «ім'я системи», «кількість працюючих процесів» тощо).

Керуюча система може отримати достовірну інформацію через операції протоколу GET, GETNEXT і GETBULK. Агент може самостійно без запиту надсилати дані, використовуючи операцію протоколу TRAP або INFORM. Управляючі системи можуть також відправляти конфігураційні оновлення або контролюючі запити, використовуючи операцію SET для безпосереднього управління системою. Операції конфігурування та управління використовуються тільки тоді, коли потрібні зміни у мережній інфраструктурі. Операції моніторингу зазвичай виконуються на регулярній основі.

Змінні, доступні через SNMP, організовані в ієрархії. Ці ієрархії та інші метадані (такі як тип і опис змінної) описуються Базами Керуючої Інформації' (Management Information Bases (MIBs)).

SNMP не визначає, яку інформацію (які змінні) керована система повинна надавати. Навпаки, SNMP використовує розширювану модель, в якій доступна інформація визначається Базами Керуючої Інформації (MIB – Management Information Base).

Бази Керуючої Інформації описують структуру керуючої інформації пристроїв. Вони використовують ієрархічний адресний простір імен, що містить унікальний ідентифікатор об'єкта (object identifier (OID)).

Грубо кажучи, кожен унікальний ідентифікатор об'єкта ідентифікує змінну, яка може бути прочитана чи встановлена через SNMP. MIB'и використовують нотацію, визначену в ASN.1.

Ієрархія MIB може бути зображена як дерево з безіменним коренем, рівні якого приписані різними організаціями. На найвищому рівні MIB OID'и належать різним організаціям, що займаються стандартизацією, в той час як на нижчих рівнях OID'и виділяються асоційованими організаціями. Ця модель забезпечує управління на всіх шарах мережної моделі OSI, адже MIB'и можуть бути визначені для будь-яких типів даних і операцій.

Керований об'єкт – це одна з будь-якого числа характеристик, специфічних для керованого пристрою. Керований об'єкт включає в себе один або більше екземплярів об'єкта (що ідентіфікуються за OID), які насправді є змінними.

Існує два типи керованих об'єктів: Скалярні об'єкти визначають єдиний екземпляр об'єкта; Табличні об'єкти визначають множинні, пов'язані екземпляри об'єктів які групуються в таблицях MIB.

Прикладом керованого об'єкта може бути atInput, який є скалярним об'єктом що містить єдиний екземпляр об'єкта, ціле число, яке показує загальну кількість вхідних пакетів AppleTalk на мережний інтерфейс маршрутизатора.

Ідентифікатор об'єкта (OID) унікально ідентифікує керований об'єкт в ієрархії MIB.

В телекомунікаціях і комп'ютерних мережах, ASN.1 є стандартною гнучкою нотацією для опису структур даних, що служать для кодування, передачі і декодування даних. ASN.1 являє собою набір правил для опису структури об'єктів, незалежних від специфічних для обладнання методик кодування, і формальну нотацію, яка дозволяє уникнути неоднозначностей.

ASN.1 це єдиний ISO та ITU-T стандарт, спочатку визначений у 1984 році як частина стандарту CCITT X.409: 1984. Пізніше, в 1988 році, завдяки його широкому застосуванню, він був виділений в окремий стандарт X.208. Значно переглянута версія 1995року описана в X.680.

Адаптована підмножина ASN.1 Структура Керуючої Інформації (SMI) описана в протоколі SNMP для визначення наборів пов'язаних MIB об'єктів, також званих MIB модулями.

SNMP працює на прикладному рівні TCP/IP (сьомий рівень моделі OSI). Агент SNMP отримує запити по UDP-порту 161. Менеджер може посилати запити з будь-якого доступного порту джерела на порт агента. Відповідь агента буде відправлений назад на порт джерела на менеджері. Менеджер отримує повідомлення (Traps і InformRequests) по порту 162. Агент може генерувати

повідомлення з будь-якого доступного порту. При використанні TLS або DTLS запити виходять по порту 10161, а пастки відправляються на порт 10162.

У SNMPv1 зазначено п'ять основних протокольних одиниць обміну (protocol data units - PDU). Ще дві PDU, GetBulkRequest і InformRequest, були введені в SNMPv2 і перенесені в SNMPv3.

Нижче перераховані сім протокольних одиниць обміну SNMP:

1. GetRequest. Запит від менеджера до об'єкту для отримання значення змінної або списку змінних. Необхідні змінні вказуються в полі variable bindings (розділ поля values при цьому не використовується). Отримання значень зазначеної змінної повинно бути виконано агентом як Атомарна операція. Менеджеру буде повернений Response (відповідь) з поточними значеннями.

2. SetRequest. Запит від менеджера до об'єкту для зміни змінної або списку змінних. Зв'язані змінні вказуються в тілі запиту. Зміни всіх зазначених змінних повинні бути виконані агентом як атомарна операція. Менеджеру буде повернений Response з (поточними) новими значеннями змінних.

3. GetNextRequest. Запит від менеджера до об'єкту для виявлення доступних змінних і їх значень. Менеджеру буде повернений Response зі зв'язаними змінними для змінної, яка є наступною в базі MIB в лексіграфіческом порядку. Обхід всієї бази MIB агента може бути проведений ітераційним використанням GetNextRequest, починаючи з OID 0. Рядки таблиці можуть бути прочитані, якщо вказати в запиті OID-и колонок в пов'язаних змінних.

4. GetBulkRequest. Покращена версія GetNextRequest. Запит від менеджера до об'єкту для численних ітерацій GetNextRequest. Менеджеру буде повернений Response з декількома пов'язаними змінними, обійденими починаючи зі пов'язаної змінної (змінних) в запиті. Специфічні для PDU поля non-repeaters і max-repetitions використовуються для контролю за поведінкою відповіді. GetBulkRequest був введений в SNMPv2.

5. Response. Повертає зв'язані змінні і значення від агента менеджеру для GetRequest, SetRequest, GetNextRequest, GetBulkRequest і InformRequest.

Повідомлення про помилки забезпечуються полями статусу помилки і індексу помилки.

Ця одиниця використовується як відповідь і на Get-, і на Set -запити, в SNMPv1 називається GetResponse.

6. Trap. Асинхронне повідомлення від агента - менеджеру. Включає в себе поточне значення sysUpTime, OID, що визначає тип trap (пастки), і необов'язкові зв'язані змінні. Адресація одержувача для пасток визначається за допомогою змінних trap-конфігурації в базі MIB. Формат trap-повідомлення був змінений в SNMPv2 і PDU перейменували в SNMPv2-Trap.

7. InformRequest. Асинхронне повідомлення від менеджера менеджеру або від агента менеджеру. Повідомлення від менеджера менеджеру були можливі вже в SNMPv1 (за допомогою Trap), але SNMP зазвичай працює на протоколі UDP, в якому доставка повідомлень не гарантована, і не повідомляється про втрачені пакетах. InformRequest виправляє це зворотним відправленням підтвердження про отримання. Одержувач відповідає Response-му, що повторює всю інформацію з InformRequest. Цей PDU був введений в SNMPv2.

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою наступного алгоритму захисту інформації DSA.

Відправник і одержувач електронного документа використовують при обчисленні великі цілі числа: G і P – прості числа, L біт кожне ($512 < L < 1024$); q – простої число довжиною 160 біт (дільник числа (P-1). Числа G, P, q є відкритими й можуть бути загальними для всіх користувачів мережі.

Відправник вибирає випадкове ціле число X, $1 < X < q$. Число X є секретним ключем відправника для формування електронного цифрового підпису. Потім відправник обчислює значення $Y = G^X \bmod P$.

Число Y є відкритим ключем для перевірки підпису відправника. Число Y

передається всім одержувачам документів.

Цей алгоритм також передбачає використання однобічної функції гешування h(-).

У стандарті DSS визначений алгоритм безпечного гешування SHA. Для того щоб підписати документ M, відправник гешує його в ціле геш-значення m: $m = h(M)$, $1 < m < q$, потім генерує випадкове ціле число K, $1 < K < q$, і обчислює число r: $r = (G^K \bmod P) \bmod q$. Потім відправник обчислює за допомогою секретного ключа X ціле число s:

$$s = \frac{m + r * X}{K} \bmod q \ .$$

Пара чисел r і s утворить цифровий підпис $S = (r, s)$ під документом M. Таким чином, підписане повідомлення являє собою трійку чисел $[M,r,s]$. Одержувач підписаного повідомлення $[M, r, s]$ перевіряє виконання умов $0 < r < q$, $0 < s < q$ і відкидає підпис, якщо хоча б одна із цих умов не виконана. Потім одержувач обчислює значення $w = 1/s \bmod q$, геш-значення $m = h(M)$ і числа $u_1 = (m*w) \bmod q$, $u_2 = (r*w) \bmod q$.

Далі одержувач за допомогою відкритого ключа Y обчислює значення $v = ((G^{U1} * Y^{U2}) \bmod P) \bmod q$ і перевіряє виконання умови $v = r$.

Якщо умова $v = r$ виконується, тоді підпис $S = (r,s)$ під документом M визнається одержувачем справжнім.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Верхнього меню: Файл; Фільтри; Звіти; Налаштування; Довідка.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Параметрів з підрозділами:Контролер; Параметри; Налаштування; Журнал подій; Формати по замовчанню; Звіти.

– Вікна виведення результату роботи системи у графічному вигляді.

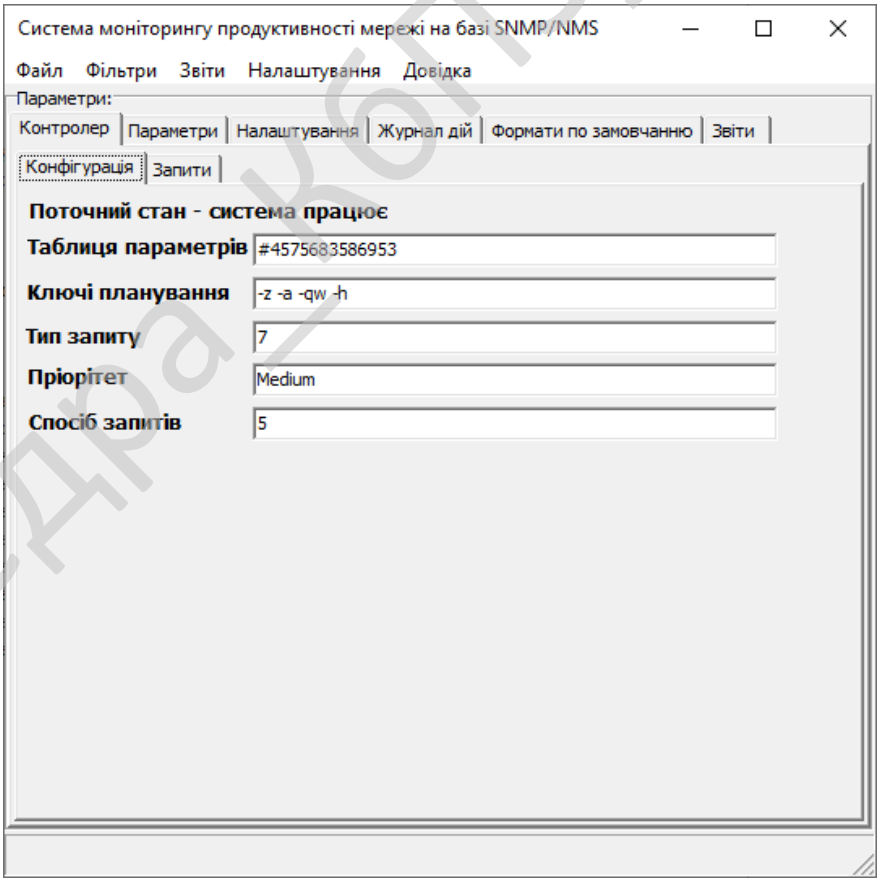

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

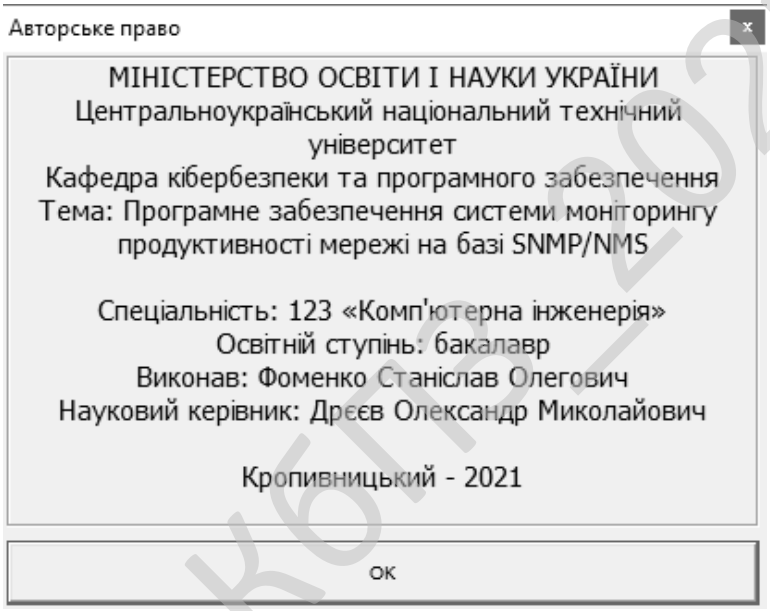

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи моніторингу продуктивності мережі на базі SNMP/NMS.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем моніторингу продуктивності мережі на базі SNMP/NMS.

– Досліджена система моніторингу продуктивності мережі на базі SNMP/NMS.

– На основі отриманих результатів досліджень створена програмна реалізація системи моніторингу продуктивності мережі на базі SNMP/NMS.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання моніторингу продуктивності мережі на базі SNMP/NMS.

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 Builder C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи моніторингу продуктивності мережі на базі SNMP/NMS. 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.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм DSA.

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. 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. Смірнова Т.В., Дрєєв О.М., «Інформаційна технологія оптимізації технологічного процесу відновлення та зміцнювання поверхонь валів зі сталі як хмарний сервіс» у *Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка*. – Х. : Вид. Рожко С.Г. 2019. С. 92-107.

45. Т.В.Смірнова, Л.І. Поліщук, «Дослідження хмарних технологій як сервісів для системи інженерних розрахунків» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. С. 106-121.

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

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

48. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів»,

Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура № 1 (11). с. 48-57, 2019.

49. Т. В. Смирнова, А. А. Смирнов, А. Н. Дреев, А. В. Дудан, «Оптимизация технологического процесса восстановления и упрочнения поверхностей с заданными характеристиками в виде облачного сервиса», Вестник Полоцкого государственного университета. Серия В, Промышленность. Прикладные науки. Республика Беларусь - 2020. - № 3. - С. 50-61.Режим доступу: http://elib.psu.by:8080/handle/123456789/24988

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

51. Т.В.Смірнова, «Формалізація та реалізація структури технологічного процесу електродугового напилення для оптимізаційної експертної системи», *Технічні науки та технології*. № 1(19). С. 104-113. 2020. (Категорія Б)

52. T. Smirnova, I. Smirnov, A. Lopata, L. Lopata «Improvement of functional properties of gas-thermal coatings by electro-contact treatment», *Problems of Tribology*, Vol. 25, № 1/95, P. 41-48. 2020.

53. Т.В. Смірнова «Формування евристичних правил, бази знань та формалізація структури й правил технологічного процесу для оптимізаційної хмарної інформаційної системи», *Системи управління, навігації та зв'язку*, № 2 (60). с. 101-104, 2020. (Категорія Б)

54. Т.В. Смірнова, О.А. Смірнов, О.М. Дрєєв, С.А. Смірнов, «Використання хмарних експертних систем в сфері інформаційного забезпечення обробки поверхні деталей», *Комп'ютерна інженерія і кібербезпека: досягнення та інновації*, м. Кропивницький. 27-29 листопада, 2018, с. 111-113

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи моніторингу продуктивності мережі на базі SNMP/NMS.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 204-02 від 28.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи моніторингу продуктивності мережі на базі SNMP/NMS.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи 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 моніторингу продуктивності мережі на базі SNMP/NMS;

– 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 XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

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 XP/Vista/7/8/10.

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 Builder C++.

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 – 71 аркуш.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

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.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2020 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 5.06.2020 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Дрєєв О.М.

Програмне забезпечення системи моніторингу продуктивності мережі на базі SNMP/NMS

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 68

Літера: РП

Кропивницький – 2021 року

resalloc.cpp – Розподіл ресурсів у системі

```
// системний файл: resalloc.cpp


#include "errno.h"

#include "tsys.h"
#include "resalloc.h"

using namespace SNMP_NMS_Application_Centric_Infrastructure;

//*****************************************
//* Об'єкт ресурсу                        *
//*****************************************
Res::Res( )
{
#if !__GLIBC_PREREQ(2,4)
    wThr = 0;
#endif
    if(pthread_rwlock_init(&rwc,NULL))
      throw TError("ResAlloc",_("Помилка відкриття семафору!"));
}

Res::~Res( )
{
    pthread_rwlock_wrlock(&rwc);
    pthread_rwlock_destroy(&rwc);
}

void Res::resRequestW( unsigned short tm )
{
    int rez = 0;
#if !__GLIBC_PREREQ(2,4)
    //EDEADLK імітація
    if(wThr && wThr == pthread_self()) rez == EDEADLK;
    else
#endif
    if(!tm) rez = pthread_rwlock_wrlock(&rwc);
    else
    {
      timespec wtm;
      clock_gettime(CLOCK_REALTIME,&wtm);
      wtm.tv_nsec += 1000000*(tm%1000);
      wtm.tv_sec += tm/1000 + wtm.tv_nsec/1000000000; wtm.tv_nsec =
wtm.tv_nsec%1000000000;
      rez = pthread_rwlock_timedwrlock(&rwc,&wtm);
    }
    if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує заблокувати
потік!"));
    else if(tm && rez == ETIMEDOUT) throw TError("ResAlloc",_("Ресурс не
відповідає!"));
#if !__GLIBC_PREREQ(2,4)
    wThr = pthread_self();
#endif
}

bool Res::resTryW( )
{
    int rez = pthread_rwlock_trywrlock(&rwc);
    if(rez == EBUSY) return false;
    else if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує
заблокувати потік!"));
    return true;
}
```

```
void Res::resRequestR( unsigned short tm )
{
    int rez = 0;
#if !__GLIBC_PREREQ(2,4)
    //EDEADLK імітація
    if(wThr && wThr == pthread_self()) rez == EDEADLK;
    else
#endif
    if(!tm) rez = pthread_rwlock_rdlock(&rwc);
    else
    {
      timespec wtm;
      clock_gettime(CLOCK_REALTIME,&wtm);
      wtm.tv_nsec += 1000000*(tm%1000);
      wtm.tv_sec += tm/1000 + wtm.tv_nsec/1000000000; wtm.tv_nsec =
wtm.tv_nsec%1000000000;
      rez = pthread_rwlock_timedrdlock(&rwc,&wtm);
    }
    if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує заблокувати
потік!"));
    else if(tm && rez == ETIMEDOUT) throw TError("ResAlloc",_("Ресурс не
відповідає!"));
}

bool Res::resTryR( )
{
    int rez = pthread_rwlock_tryrdlock(&rwc);
    if(rez == EBUSY) return false;
    else if(rez == EDEADLK) throw TError(10,"ResAlloc",_("Ресурс пробує
заблокувати потік!"));
    return true;
}

void Res::resRelease( )
{
    pthread_rwlock_unlock(&rwc);
#if !__GLIBC_PREREQ(2,4)
    if(wThr == pthread_self()) wThr = 0;
#endif
}

//*****************************************
//* Автоматичний розподіл ресурсів *
//*****************************************
ResAlloc::ResAlloc( Res &rid ) : mId(rid), mAlloc(false)
{

}

ResAlloc::ResAlloc( Res &rid, bool write, unsigned short tm ) : mId(rid),
mAlloc(false)
{
    request(write, tm);
}

ResAlloc::~ResAlloc( )
{
    if(mAlloc) release();
}

void ResAlloc::request( bool write, unsigned short tm )
{
    if(mAlloc) release();
    mAlloc = false;
    try
    {
      if(write) mId.resRequestW(tm);
      else mId.resRequestR(tm);
      mAlloc = true;
```

```
    }catch(TError err) { if(err.cod!=10) throw; }
}

void ResAlloc::release()
{
    if(!mAlloc) return;
    mId.resRelease( );
    mAlloc = false;
}

//*****************************************
//* Рядок + ресурс для                    *
//*****************************************
ResString::ResString( const string &vl )
{
    pthread_mutex_init(&mRes, NULL);
    setVal(vl);
}

ResString::~ResString( )
{
    pthread_mutex_lock(&mRes);
    pthread_mutex_destroy(&mRes);
}

size_t ResString::size( )     { return getVal().size(); }

bool   ResString::empty( )    { return getVal().empty(); }

void ResString::setVal( const string &vl )
{
    pthread_mutex_lock(&mRes);
    str = vl;
    pthread_mutex_unlock(&mRes);
}

string ResString::getVal( )
{
    string rez;
    pthread_mutex_lock(&mRes);
    rez = str;
    pthread_mutex_unlock(&mRes);
    return rez;
}

ResString &ResString::operator=( const string &val )
{
    setVal(val);
    return *this;
}
```

tarchives.cpp – Робота з архівом

```
//WebSNMP_NMS_Application_Centric_Infrastructure системний файл: tarchives.cpp

#include <unistd.h>
#include <getopt.h>
#include <signal.h>
#include <sys/time.h>
#include <string.h>
#include <algorithm>

#include "tsys.h"
#include "tarchives.h"

#define BUF_SIZE_DEF 500
#define BUF_SIZE_MAX 100000

using namespace SNMP_NMS_Application_Centric_Infrastructure;

//************************************************
//* Підсистема архівування                       *
//************************************************

//************************************************
//* TArchiveS                                    *
//************************************************
TArchiveS::TArchiveS( ) :
    TSubSYS(SARH_ID,"Archives",true), elMess(""), elVal(""), elAval(""),
bufErr(0), mMessPer(10), prcStMess(false),
    headBuf(0), headLstread(0), mValPer(1000), mValPrior(10), prcStVal(false),
endrunReqVal(false)
{
    mAval = grpAdd("va_");

    //> архіватор повідомлення у структурі БД
    elMess.fldAdd( new TFld("ID",_("ID"),TFld::String,TCfg::Key,"20") );
    elMess.fldAdd( new TFld("MODUL",_(" Ім’я модуля
(плагіна)"),TFld::String,TCfg::Key,"20") );
    elMess.fldAdd( new
TFld("NAME",_("Ім’я"),TFld::String,TCfg::TransltText,"50") );
    elMess.fldAdd( new
TFld("DESCR",_("Дескриптор"),TFld::String,TCfg::TransltText,"200") );
    elMess.fldAdd( new TFld("START",_("Початок архіву"),TFld::Boolean,0,"1") );
    elMess.fldAdd( new TFld("CATEG",_("Категорії
повідомлень"),TFld::String,0,"100") );
    elMess.fldAdd( new TFld("LEVEL",_("Рівні
повідомлень"),TFld::Integer,0,"1","","0;7") );
    elMess.fldAdd( new TFld("ADDR",_("Адреса"),TFld::String,0,"100") );

    //> Значення архіватора у структурі БД
    elVal.fldAdd( new TFld("ID",_("ID"),TFld::String,TCfg::Key,"20") );
    elVal.fldAdd( new TFld("MODUL",_("Ім’я модуля
 (плагіна)"),TFld::String,TCfg::Key,"20") );
    elVal.fldAdd( new TFld("NAME",_(" Ім’я
"),TFld::String,TCfg::TransltText,"50") );
    elVal.fldAdd( new
TFld("DESCR",_("Дескриптор"),TFld::String,TCfg::TransltText,"200") );
    elVal.fldAdd( new TFld("START",_("Початок архіву"),TFld::Boolean,0,"1","0")
);
    elVal.fldAdd( new TFld("ADDR",_("Адреса"),TFld::String,0,"50") );
    elVal.fldAdd( new TFld("V_PER",_("Value period
(sec)"),TFld::Real,0,"12.6","1","0;1000000") );
    elVal.fldAdd( new TFld("A_PER",_("Період архівації
(sec)"),TFld::Integer,0,"4","60","0;1000") );

    //> Значення архіву у структурі БД
    elAval.fldAdd( new TFld("ID",_("ID"),TFld::String,TCfg::Key,"20") );
```

```
    elAval.fldAdd( new TFld("NAME",_(" Ім'я
"),TFld::String,TCfg::TransltText,"50") );
    elAval.fldAdd( new
TFld("DESCR",_("Дескриптор"),TFld::String,TCfg::TransltText,"200") );
    elAval.fldAdd( new TFld("START",_("Початок архіву"),TFld::Boolean,0,"1","0")
);
    elAval.fldAdd( new TFld("SrcMode",_("Режим джерела"),TFld::Integer,0,"1") );
    elAval.fldAdd( new TFld("Source",_("Джерело"),TFld::String,0,"100") );
    elAval.fldAdd( new TFld("VTYPE",_("Тип значення"),TFld::Integer,0,"1") );
    elAval.fldAdd( new TFld("BPER",_("Період буферізації
(sec)"),TFld::Real,0,"9.6","1","0;10000") );
    elAval.fldAdd( new TFld("BSIZE",_("Розмір
буферу()"),TFld::Integer,0,"6","100","0;1000000") );
    elAval.fldAdd( new TFld("BHGRD",_("Буфер у режимі грід-
системи"),TFld::Boolean,0,"1","1") );
    elAval.fldAdd( new TFld("BHRES",_("Значення буфера останім
часом"),TFld::Boolean,0,"1","0") );
    elAval.fldAdd( new TFld("ArchS",_("Процес
архівування"),TFld::String,0,"500") );

    setMessBufLen( BUF_SIZE_DEF );

    //> Створення повідомлення часу архівування
    struct sigevent sigev;
    memset(&sigev,0,sizeof(sigev));
    sigev.sigev_notify = SIGEV_THREAD;
    sigev.sigev_value.sival_ptr = this;
    sigev.sigev_notify_function = ArhMessTask;
    sigev.sigev_notify_attributes = NULL;
    timer_create(CLOCK_REALTIME,&sigev,&tmIdMess);
}

TArchiveS::~TArchiveS(  )
{
    //> Поввідомлення про закінчення архівування
    timer_delete(tmIdMess);

    //> Перерривання архівування
    if(prcStVal) SYS->taskDestroy(nodePath('.',true)+".vals", &endrunReqVal);

    //> Визволення усіх ресурсів
    nodeDelAll();
}

int TArchiveS::valPeriod( )         { return vmax(1,mValPer); }

void TArchiveS::setValPrior( int ivl )    { mValPrior = vmax(-1,vmin(99,ivl));
modif(); }

void TArchiveS::load_( )
{
    //> Завантажуємо параметри з командного рядка
    int next_opt;
    const char *short_opt="h";
    struct option long_opt[] =
    {
      {"help"    ,0,NULL,'h'},
      {NULL      ,0,NULL,0  }
    };

    optind=0,opterr=0;
    do
    {
      next_opt=getopt_long(SYS->argc,(char * const *)SYS-
>argv,short_opt,long_opt,NULL);
      switch(next_opt)
      {
          case 'h': fprintf(stdout,"%s",optDescr().c_str()); break;
          case -1 : break;
```

```
      }
    } while(next_opt != -1);

    //> Завантажуємо параметри
    setMessBufLen(
atoi(TBDS::genDBGet(nodePath()+"MessBufSize",TSYS::int2str(messBufLen())).c_str(
)) );
    setMessPeriod(
atoi(TBDS::genDBGet(nodePath()+"MessPeriod",TSYS::int2str(mMessPer)).c_str()) );
    setValPeriod(
atoi(TBDS::genDBGet(nodePath()+"ValPeriod",TSYS::int2str(mValPer)).c_str()) );
    setValPrior(
atoi(TBDS::genDBGet(nodePath()+"ValPriority",TSYS::int2str(mValPrior)).c_str())
);

    //> LidDB
    //>> Повідомлення завантаження архіватора
    string id,type;
    map<string, bool>   itReg;
    try
    {
      TConfig c_el(&elMess);
      c_el.cfgViewAll(false);
      vector<string> db_ls;

      //>> Шукаємо у БД і створюємо новий архів
      SYS->db().at().dbList(db_ls,true);
      db_ls.push_back("<cfg>");
      for(unsigned i_db = 0; i_db < db_ls.size(); i_db++)
          for(int fld_cnt=0; SYS-
>db().at().dataSeek(db_ls[i_db]+"."+subId()+"_mess_proc",nodePath()+subId()+"_me
ss_proc",fld_cnt++,c_el); )
          {
            id = c_el.cfg("ID").getS();
            type = c_el.cfg("MODUL").getS();
            if(modPresent(type) && !at(type).at().messPresent(id))
                at(type).at().messAdd(id,(db_ls[i_db]==SYS-
>workDB())?"*.*":db_ls[i_db]);
            itReg[type+"."+id] = true;
          }

      //>>> Перевіряємо для видалення з БД
        if(!SYS->selDB().empty())
        {
            vector<string> m_ls;
            modList(m_ls);
            for(unsigned i_m = 0; i_m < m_ls.size(); i_m++)
            {
                at(m_ls[i_m]).at().messList(db_ls);
                for(unsigned i_it = 0; i_it < db_ls.size(); i_it++)
                    if(itReg.find(m_ls[i_m]+"."+db_ls[i_it]) == itReg.end() &&
SYS->chkSelDB(at(m_ls[i_m]).at().messAt(db_ls[i_it]).at().DB()))
                        at(m_ls[i_m]).at().messDel(db_ls[i_it]);
            }
        }
    }catch( TError err )
    {
      mess_err(err.cat.c_str(),"%s",err.mess.c_str());
      mess_err(nodePath().c_str(),_("Повідомлення вомилки завантаження
архіватора."));
    }

    //>> Завантажуємо значення архіватора
    try
    {
      TConfig c_el(&elVal);
      c_el.cfgViewAll(false);
      vector<string> db_ls;
      itReg.clear();
```

```
      //>> Шукаємо у БД та створюємо новий архів
      SYS->db().at().dbList(db_ls,true);
      db_ls.push_back("<cfg>");
      for(unsigned i_db = 0; i_db < db_ls.size(); i_db++)
          for(int fld_cnt=0; SYS-
>db().at().dataSeek(db_ls[i_db]+"."+subId()+"_val_proc",nodePath()+subId()+"_val
_proc",fld_cnt++,c_el); )
          {
            id = c_el.cfg("ID").getS();
            type = c_el.cfg("MODUL").getS();
            if(modPresent(type) && !at(type).at().valPresent(id))
                at(type).at().valAdd(id,(db_ls[i_db]==SYS-
>workDB())?"*.*":db_ls[i_db]);
            itReg[type+"."+id] = true;
          }

      //>>> Перевіряємо для видалення з БД
        if(!SYS->selDB().empty())
        {
            vector<string> m_ls;
            modList(m_ls);
            for(unsigned i_m = 0; i_m < m_ls.size(); i_m++)
            {
                at(m_ls[i_m]).at().valList(db_ls);
                for(unsigned i_it = 0; i_it < db_ls.size(); i_it++)
                    if(itReg.find(m_ls[i_m]+"."+db_ls[i_it]) == itReg.end() &&
SYS->chkSelDB(at(m_ls[i_m]).at().valAt(db_ls[i_it]).at().DB()))
                        at(m_ls[i_m]).at().valDel(db_ls[i_it]);
            }
        }
    }catch( TError err )
    {
      mess_err(err.cat.c_str(),"%s",err.mess.c_str());
      mess_err(nodePath().c_str(),_("Помилка завантаження значення
архіватора."));
    }

    //>> Завантажуємо значення архіватора
    try
    {
      TConfig c_el(&elAval);
      c_el.cfgViewAll(false);
      vector<string> db_ls;
      itReg.clear();

      //>> Шукаємо у БД та створюємо новий архів
      SYS->db().at().dbList(db_ls,true);
      db_ls.push_back("<cfg>");
      for(unsigned i_db = 0; i_db < db_ls.size(); i_db++)
          for(int fld_cnt=0; SYS-
>db().at().dataSeek(db_ls[i_db]+"."+subId()+"_val",nodePath()+subId()+"_val",fld
_cnt++,c_el); )
          {
              id = c_el.cfg("ID").getS();
              if(!valPresent(id)) valAdd(id,(db_ls[i_db]==SYS-
>workDB())?"*.*":db_ls[i_db]);
              itReg[id] = true;
          }

      //>>> Перевіряємо для видалення з БД
        if(!SYS->selDB().empty())
        {
            valList(db_ls);
            for(unsigned i_it = 0; i_it < db_ls.size(); i_it++)
                if(itReg.find(db_ls[i_it]) == itReg.end() && SYS-
>chkSelDB(valAt(db_ls[i_it]).at().DB()))
                    valDel(db_ls[i_it]);
        }
```

```
    }catch(TError err)
    {
      mess_err(err.cat.c_str(),"%s",err.mess.c_str());
      mess_err(nodePath().c_str(),_("Помилка завантаження значення
архіватора."));
    }
}

void TArchiveS::save_( )
{
    vector<string> t_lst, o_lst;

    //> Зберігаємо параметри
    TBDS::genDBSet(nodePath()+"MessBufSize",TSYS::int2str(messBufLen()));
    TBDS::genDBSet(nodePath()+"MessPeriod",TSYS::int2str(messPeriod()));
    TBDS::genDBSet(nodePath()+"ValPeriod",TSYS::int2str(valPeriod()));
    TBDS::genDBSet(nodePath()+"ValPriority",TSYS::int2str(valPrior()));
}

void TArchiveS::valAdd( const string &iid, const string &idb )
{
    if( valPresent(iid) ) return;
    chldAdd(mAval,new TVArchive(iid,idb,&aValE()));
}

string TArchiveS::optDescr(  )
{
    char buf[STR_BUF_LEN];
    snprintf(buf,sizeof(buf),_(
      "======================== Підсистема \"Архів\" Опції
==================\n"
      "------------ Параметри частини '%s' у кофігураційний файл -----------\n"
      "MessBufSize   <items>       Повідомлення розміру буферу.\n"
      "MessPeriod    <sec>         Повідомлення періоду архівування.\n"
      "ValPeriod     <msec>        Значення періоду архівування.\n"
      "ValPriority   <level>       Значення завдання пріорітетного рівня.\n"
      "MaxReqMess    <items>       Повідомлення максимального запиту.\n"
      "MaxReqVals    <items>       Значення  максимального запиту.\n\n"
      ),nodePath().c_str());

    return buf;
}

void TArchiveS::subStart( )
{
    mess_info(nodePath().c_str(),_("Запускаємо підсистеми."));

    SubStarting = true;

    vector<string> t_lst, o_lst;

    modList(t_lst);
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      AutoHD<TTipArchivator> mod = modAt(t_lst[i_t]);

      //> Повідомлення про початок роботи архіватора
      mod.at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TMArchivator> mess = mod.at().messAt(o_lst[i_o]);
          if( /*!mess.at().startStat() &&*/ mess.at().toStart() )
            try{ mess.at().start(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Повідомлення про помилку роботи
архіватора."),o_lst[i_o].c_str());
            }
```

```
      }
      //> Значення початку роботи архіватора
      mod.at().valList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TVArchivator> val = mod.at().valAt(o_lst[i_o]);
          if( /*!val.at().startStat() &&*/ val.at().toStart() )
            try{ val.at().start(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Помилка початку роботи
архіватора."),val.at().workId().c_str());
            }
      }
    }

    //> Значення початку роботи архіватора
    valList(o_lst);
    for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
    {
      AutoHD<TVArchive> aval = valAt(o_lst[i_o]);
      if( /*!aval.at().startStat() &&*/ aval.at().toStart() )
          try{ aval.at().start(); }
          catch(TError err)
          {
            mess_err(err.cat.c_str(),"%s",err.mess.c_str());
            mess_err(nodePath().c_str(),_("Помилка початку роботи
архіватора"),o_lst[i_o].c_str());
          }
    }

    //> Повідомлення про початок роботи інтервального таймера
    struct itimerspec itval;
    itval.it_interval.tv_sec = itval.it_value.tv_sec = messPeriod();
    itval.it_interval.tv_nsec = itval.it_value.tv_nsec = 0;
    timer_settime(tmIdMess, 0, &itval, NULL);

    //> Значення початку роботи завдань
    if(!prcStVal) SYS->taskCreate(nodePath('.',true)+".vals", valPrior(),
TArchiveS::ArhValTask, this);

    TSubSYS::subStart( );

    SubStarting = false;
}

void TArchiveS::subStop( )
{
    mess_info(nodePath().c_str(),_("Зупинка підсистеми."));

    TSubSYS::subStop( );

    vector<string> t_lst, o_lst;

    //> Зупинка інтервального таймера для періодичних потоків, для створення
повідомлення архівації структур;
    itval.it_interval.tv_sec = itval.it_interval.tv_nsec =
      itval.it_value.tv_sec = itval.it_value.tv_nsec = 0;
    timer_settime(tmIdMess, 0, &itval, NULL);
    if(TSYS::eventWait( prcStMess, false, nodePath()+"mess_stop",10))
      throw TError(nodePath().c_str(),_("Архівація повідомлень потоків не може
бути зупинена!"));

    //> Значення зупинки роботи завдань
    if(prcStVal) SYS->taskDestroy(nodePath('.',true)+".vals", &endrunReqVal);

    //> Виклик останього повідомлення архіватора
    sigval obj; obj.sival_ptr = this;
```

```
    ArhMessTask(obj);

    //> Зупинка архіватора
    modList(t_lst);
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      AutoHD<TTipArchivator> mod = modAt(t_lst[i_t]);
      //Значення зупинки архіватора
      mod.at().valList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TVArchivator> val = mod.at().valAt(o_lst[i_o]);
          if( val.at().startStat() )
            try{ val.at().stop(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Значення архіватора '%s' stop
error."),o_lst[i_o].c_str());
            }
      }
      // Повідомлення про зупинку архіватора
      mod.at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          AutoHD<TMArchivator> mess = mod.at().messAt(o_lst[i_o]);
          if( mess.at().startStat() )
            try{ mess.at().stop(); }
            catch(TError err)
            {
                mess_err(err.cat.c_str(),"%s",err.mess.c_str());
                mess_err(nodePath().c_str(),_("Повідомлення про помилку зупинки
архіватора"),o_lst[i_o].c_str());
            }
      }
    }

    //> Значення зупинки архіватора
    valList(o_lst);
    for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
    {
      AutoHD<TVArchive> aval = valAt(o_lst[i_o]);
      if( aval.at().startStat() )
          try{ aval.at().stop(); }
          catch(TError err)
          {
            mess_err(err.cat.c_str(),"%s",err.mess.c_str());
            mess_err(nodePath().c_str(),_("Значення помилки зупинки
архіватора"),o_lst[i_o].c_str());
          }
    }
}

void TArchiveS::messPut( time_t tm, int utm, const string &categ, int8_t level,
const string &mess )
{
    //> Відправляємо повідомлення у буфер
    ResAlloc res(mRes,true);
    mBuf[headBuf].time  = tm;
    mBuf[headBuf].utime = utm;
    mBuf[headBuf].categ = categ;
    mBuf[headBuf].level = (TMess::Type)abs(level);
    mBuf[headBuf].mess  = mess;
    if( ++headBuf >= mBuf.size() ) headBuf = 0;
    //> Перервіряємо, чи це не повідомлення архіватора
    if( headBuf == headLstread )
    {
      if( !(bufErr&0x01) )
      {
```

```
            bufErr |= 0x01;
            res.release();
            mess_err(nodePath().c_str(),_("Буфер заповнений. Останнє
повідомлення!"));
            res.request(true);
        }
        if( ++headLstread >= mBuf.size() ) headLstread = 0;
      }
      //> Перевіряємо швидкість заповнення буфера.
      else if( headBuf-headLstread > messBufLen( )/2 )
      {
        if( !(bufErr&0x02) )
        {
            bufErr |= 0x02;
            res.release();
            mess_warning(nodePath().c_str(),_("Повідомлення про т, що швидкість
заповнення буфера дуже висока!"));
            res.request(true);
        }
      }
      else bufErr = 0;

      //> Обробка тривог. Для рівня менше 0 тривоги встановлено
      map<string,TMess::SRec>::iterator p;
      if( level < 0 ) mAlarms[categ] =
TMess::SRec(tm,utm,categ,(TMess::Type)abs(level),mess);
      else if( (p=mAlarms.find(categ)) != mAlarms.end() ) mAlarms.erase(p);
}

void TArchiveS::messPut( const vector<TMess::SRec> &recs )
{
      for(unsigned i_r = 0; i_r < recs.size(); i_r++)
        messPut(recs[i_r].time,recs[i_r].utime,recs[i_r].categ,recs[i_r].level,rec
s[i_r].mess);
}

void TArchiveS::messGet( time_t b_tm, time_t e_tm, vector<TMess::SRec> & recs,
const string &category, int8_t level, const string &arch, time_t upTo )
{
      recs.clear();

      ResAlloc res(mRes,false);
      if(!upTo) upTo = time(NULL)+STD_INTERF_TM;
      TRegExp re(category, "p");

      //> Отримуємо записи з буфера
      unsigned i_buf = headBuf;
      while(level >= 0 && (!arch.size() || arch==BUF_ARCH_NM) && time(NULL) <
upTo)
      {
        if(mBuf[i_buf].time >= b_tm && mBuf[i_buf].time != 0 && mBuf[i_buf].time
<= e_tm &&
            mBuf[i_buf].level >= level && re.test(mBuf[i_buf].categ))
            recs.push_back(mBuf[i_buf]);
        if(++i_buf >= mBuf.size()) i_buf = 0;
        if(i_buf == headBuf) break;
      }

      //> Отримуємо записи з архівів
      vector<string> t_lst, o_lst;
      modList(t_lst);
      for(unsigned i_t = 0; level >= 0 && i_t < t_lst.size(); i_t++)
      {
        at(t_lst[i_t]).at().messList(o_lst);
        for(unsigned i_o = 0; i_o < o_lst.size() && time(NULL) < upTo; i_o++)
        {
            AutoHD<TMArchivator> archtor = at(t_lst[i_t]).at().messAt(o_lst[i_o]);
            if(archtor.at().startStat() && (!arch.size() ||
arch==archtor.at().workId()))
```

```
            archtor.at().get(b_tm,e_tm,recs,category,level);
      }
    }

    //> Запит процесу тривоги
    if(level < 0)
    {
      vector< pair<int64_t,TMess::SRec* > > mb;
      for(map<string,TMess::SRec>::iterator p = mAlarms.begin(); p !=
mAlarms.end() && time(NULL) < upTo; p++)
        if((p->second.time >= b_tm || b_tm == e_tm) && p->second.time <= e_tm
&&
            p->second.level >= abs(level) && re.test(p->second.categ))
          mb.push_back(pair<int64_t,TMess::SRec* >(FTM(p->second),&p-
>second));
      sort(mb.begin(),mb.end());
      for(unsigned i_b = 0; i_b < mb.size(); i_b++)
recs.push_back(*mb[i_b].second);
    }
}

time_t TArchiveS::messBeg( const string &arch )
{
    time_t rez = 0;
    ResAlloc res(mRes,false);
    if(arch.empty() || arch == BUF_ARCH_NM)
    {
      unsigned i_buf = headBuf;
      while(!arch.size() || arch == BUF_ARCH_NM)
      {
          rez = rez ? vmin(rez,mBuf[i_buf].time) : mBuf[i_buf].time;
          if(++i_buf >= mBuf.size()) i_buf = 0;
          if(i_buf == headBuf) break;
      }
      if( !arch.empty() ) return rez;
    }

    //- Отримуємо записи з архівів -
    vector<string> t_lst, o_lst;
    modList(t_lst);
    AutoHD<TMArchivator> archtor;
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      at(t_lst[i_t]).at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          archtor = at(t_lst[i_t]).at().messAt(o_lst[i_o]);
          if(archtor.at().startStat() && (!arch.size() ||
arch==archtor.at().workId()))
            rez = rez ? vmin(rez,archtor.at().begin()) : archtor.at().begin();
      }
    }

    return rez;
}

time_t TArchiveS::messEnd( const string &arch )
{
    time_t rez = 0;
    ResAlloc res(mRes,false);
    if(arch.empty() || arch == BUF_ARCH_NM)
    {
      unsigned i_buf = headBuf;
      while(!arch.size() || arch == BUF_ARCH_NM)
      {
          rez = rez ? vmax(rez,mBuf[i_buf].time) : mBuf[i_buf].time;
          if(++i_buf >= mBuf.size()) i_buf = 0;
          if(i_buf == headBuf) break;
      }
```

```
      if(!arch.empty()) return rez;
    }

    //> Отримуємо записи з архівів
    vector<string> t_lst, o_lst;
    modList(t_lst);
    AutoHD<TMArchivator> archtor;
    for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
    {
      at(t_lst[i_t]).at().messList(o_lst);
      for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
      {
          archtor = at(t_lst[i_t]).at().messAt(o_lst[i_o]);
          if(archtor.at().startStat() && (!arch.size() ||
arch==archtor.at().workId()))
            rez = rez ? vmax(rez,archtor.at().end()) : archtor.at().end();
      }
    }

    return rez;
}

void TArchiveS::setMessBufLen(unsigned len)
{
    ResAlloc res(mRes,true);
    len = vmin(BUF_SIZE_MAX,vmax(BUF_SIZE_DEF,len));
    while(mBuf.size() > len)
    {
      mBuf.erase(mBuf.begin() + headBuf);
      if(headBuf >= mBuf.size()) headBuf = 0;
      if(headLstread >= mBuf.size())     headLstread = mBuf.size()-1;
    }
    while(mBuf.size() < len) mBuf.insert(mBuf.begin() + headBuf, TMess::SRec());
    modif();
}

void TArchiveS::setActValArch( const string &id, bool val )
{
    unsigned i_arch;

    ResAlloc res(vRes,true);
    for( i_arch = 0; i_arch < actUpSrc.size(); i_arch++ )
      if( actUpSrc[i_arch].at().id() == id ) break;

    if( val && i_arch >= actUpSrc.size() )
      actUpSrc.push_back(valAt(id));
    if( !val && i_arch < actUpSrc.size() )
      actUpSrc.erase(actUpSrc.begin()+i_arch);
}

void TArchiveS::setMessPeriod( int ivl )
{
    mMessPer = ivl;
    modif();

    if( subStartStat( ) )
    {
      struct itimerspec itval;
      itval.it_interval.tv_sec = itval.it_value.tv_sec = mMessPer;
      itval.it_interval.tv_nsec = itval.it_value.tv_nsec = 0;
      timer_settime(tmIdMess, 0, &itval, NULL);
    }
}

void TArchiveS::ArhMessTask( union sigval obj )
{
    TArchiveS &arh = *(TArchiveS *)obj.sival_ptr;
    if( arh.prcStMess )  return;
    arh.prcStMess = true;
```

```
    //> Читаємо повідомлення з буфера
    if( arh.headLstread != arh.headBuf )
      try
      {
          ResAlloc res(arh.mRes,false);

          //>> Беремо нове повідомлення
          unsigned new_headLstread = arh.headBuf;
          unsigned i_m = arh.headLstread;
          vector<TMess::SRec> o_mess;
          while( i_m != new_headLstread )
          {
            o_mess.push_back(arh.mBuf[i_m]);
            if( ++i_m >= arh.mBuf.size() ) i_m = 0;
          }
          arh.headLstread = new_headLstread;

          res.release();

          //>> Архівуємо
          vector<string> t_lst, o_lst;
          arh.modList(t_lst);
          for(unsigned i_t = 0; i_t < t_lst.size(); i_t++)
          {
            arh.at(t_lst[i_t]).at().messList(o_lst);
            for(unsigned i_o = 0; i_o < o_lst.size(); i_o++)
                if(arh.at(t_lst[i_t]).at().messAt(o_lst[i_o]).at().startStat())
                  arh.at(t_lst[i_t]).at().messAt(o_lst[i_o]).at().put(o_mess);
          }
      }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(arh.nodePath().c_str(),_("Помилка читання повідомлення з
буфера."));
      }

    arh.prcStMess = false;
}

void *TArchiveS::ArhValTask( void *param )
{
    TArchiveS &arh = *(TArchiveS *)param;
    arh.endrunReqVal = false;
    arh.prcStVal = true;

    while( !arh.endrunReqVal )
    {
      int64_t work_tm = SYS->curTime();

      arh.vRes.resRequestR( );
      for(unsigned i_arh = 0; i_arh < arh.actUpSrc.size(); i_arh++)
          try
          {
            if( work_tm/arh.actUpSrc[i_arh].at().period() >
arh.actUpSrc[i_arh].at().end()/arh.actUpSrc[i_arh].at().period() )
                arh.actUpSrc[i_arh].at().getActiveData();
          }
          catch(TError err)
          { mess_err(err.cat.c_str(),"%s",err.mess.c_str()); }
      arh.vRes.resRelease( );

      TSYS::taskSleep((int64_t)arh.valPeriod()*1000000);
    }

    arh.prcStVal = false;

    return NULL;
```

```
}

TVariant TArchiveS::objFuncCall( const string &iid, vector<TVariant> &prms,
const string &user )
{
    // Array messGet(int btm, int etm, string cat = "", int lev = 0, string arch
= ""); - запит системних повідомлень, для часу з <btm>
    //       до <etm> для категорії <cat>, рівня <lev> та архіву <arch>
    //  btm - час початку
    //  etm - час закінчення
    //  cat - категорія повідомлення
    //  lev - рівні повідомлень
    //  arch - архівація повідомлення
    if( iid == "messGet" && prms.size() >= 2 )
    {
      vector<TMess::SRec> recs;
      messGet( prms[0].getI(), prms[1].getI(), recs, ((prms.size()>=3) ?
prms[2].getS() : string("")),
          ((prms.size()>=4) ? prms[3].getI() : 0), ((prms.size()>=5) ?
prms[4].getS() : string("")) );
      TArrayObj *rez = new TArrayObj();
      for(unsigned i_m = 0; i_m < recs.size(); i_m++)
      {
          TVarObj *am = new TVarObj();
          am->propSet("tm",(int)recs[i_m].time);
          am->propSet("utm",recs[i_m].utime);
          am->propSet("categ",recs[i_m].categ);
          am->propSet("level",recs[i_m].level);
          am->propSet("mess",recs[i_m].mess);
          rez->propSet(TSYS::int2str(i_m),am);
      }
      return rez;
    }

    return TCntrNode::objFuncCall(iid,prms,user);
}

void TArchiveS::cntrCmdProc( XMLNode *opt )
{
    string a_path = opt->attr("path");
    //> Сервіс управління процесами
    if(a_path == "/serv/mess")            //Повідомлення доступу
    {
      if(ctrChkNode(opt,"info",RWRWRW,"root",SARH_ID,SEC_RD))
      //Інформаційні повідомлення
      {
          string arch = opt->attr("arch");
          opt->setAttr("end",TSYS::uint2str(messEnd(arch)));
          opt->setAttr("beg",TSYS::uint2str(messBeg(arch)));
      }
      else if(ctrChkNode(opt,"get",RWRWRW,"root",SARH_ID,SEC_RD))//Запит
значення дати
      {
          time_t tm      = strtoul(opt->attr("tm").c_str(),0,10);
          time_t tm_grnd = strtoul(opt->attr("tm_grnd").c_str(),0,10);
          string arch    = opt->attr("arch");
          string cat     = opt->attr("cat");
          int    lev     = atoi(opt->attr("lev").c_str());
          vector<TMess::SRec> rez;
          messGet( tm_grnd, tm, rez, cat, (TMess::Type)lev, arch );
          for(unsigned i_r = 0; i_r < rez.size(); i_r++)
            opt->childAdd("el")->
                setAttr("time",TSYS::uint2str(rez[i_r].time))->
                setAttr("utime",TSYS::uint2str(rez[i_r].utime))->
                setAttr("cat",rez[i_r].categ)->
                setAttr("lev",TSYS::int2str(rez[i_r].level))->
                setText(rez[i_r].mess);
      }
      return;
```

```
    }

    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TSubSYS::cntrCmdProc(opt);
      ctrMkNode("grp",opt,-1,"/br/va_",_("Архів
значень"),RWRWR_,"root",SARH_ID,2,"idm","1","idSz","20");
      if(ctrMkNode("area",opt,1,"/m_arch",_("Архів
повідомлень"),R_R_R_,"root",SARH_ID))
      {
          ctrMkNode("fld",opt,-1,"/m_arch/size",_("Повідомлення розміру
буферу"),RWRWR_,"root",SARH_ID,2,"tp","dec","min",TSYS::int2str(BUF_SIZE_DEF).c_
str());
          ctrMkNode("fld",opt,-1,"/m_arch/per",_("Період архівування
(s)"),RWRWR_,"root",SARH_ID,1,"tp","dec");
          if(ctrMkNode("area",opt,-1,"/m_arch/view",_("Перегляд
повідомлень"),R_R___,"root",SARH_ID))
          {
            ctrMkNode("fld",opt,-
1,"/m_arch/view/tm",_("Time"),RWRW__,"root",SARH_ID,1,"tp","time");
            ctrMkNode("fld",opt,-1,"/m_arch/view/size",_("Розмір
(s)"),RWRW__,"root",SARH_ID,1,"tp","dec");
            ctrMkNode("fld",opt,-1,"/m_arch/view/cat",_("Зразок
категорії"),RWRW__,"root",SARH_ID,2,"tp","str","help",
                _("Шаблон категорії повідомлень або регулярне вираження.\n"
                    "Використовуйте тимчасові символи для групового
виділення:\n  '*' - будь-які підрядки;\n  '?' - будь-які символи.\n"
                  "Регулярне вираження складається з символів'/'
(/mod_(System|LogicLev)/)."));
            ctrMkNode("fld",opt,-
1,"/m_arch/view/lvl",_("Рівень"),RWRW__,"root",SARH_ID,4,"tp","dec","min","-
7","max","7",
                "help",_("Отримуємо повідомлення для рівня більшого і
дорівнюючого цьому."));
            ctrMkNode("fld",opt,-
1,"/m_arch/view/archtor",_("Архіватор"),RWRW__,"root",SARH_ID,4,"tp","str","dest
","select","select","/m_arch/lstAMess",
                "help",_("Архівація повідомлення.\n Не встановлюємо архіватор
для процесів у буфері та інших архіваторів.\Встановлюємо '<buffer>' для процесів
у буфері ."));
            if(ctrMkNode("table",opt,-
1,"/m_arch/view/mess",_("Messages"),R_R___,"root",SARH_ID))
            {
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0",_("Time"),R_R___,"root",SARH_ID,1,"tp","time");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0a",_("mcsec"),R_R___,"root",SARH_ID,1,"tp","dec");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/1",_("Category"),R_R___,"root",SARH_ID,1,"tp","str");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/2",_("Lev."),R_R___,"root",SARH_ID,1,"tp","dec");
                ctrMkNode("list",opt,-
1,"/m_arch/view/mess/3",_("Message"),R_R___,"root",SARH_ID,1,"tp","str");
            }
          }
      }
      if(ctrMkNode("area",opt,2,"/v_arch",_("Архів
значеньь"),R_R_R_,"root",SARH_ID))
      {
          ctrMkNode("fld",opt,-1,"/v_arch/per",_("Беремо дані періоду
(ms)"),RWRWR_,"root",SARH_ID,1,"tp","dec");
          ctrMkNode("fld",opt,-1,"/v_arch/prior",_("Беремо дані завдання
пріорітетного рівня "),RWRWR_,"root",SARH_ID,1,"tp","dec");
          ctrMkNode("fld",opt,-
1,"/v_arch/nmb",_("Number"),R_R_R_,"root",SARH_ID,1,"tp","str");
          ctrMkNode("list",opt,-1,"/v_arch/archs",_("Архів
значень"),RWRWR_,"root",SARH_ID,5,"tp","br","idm","1","s_com","add,del","br_pref
","va_","idSz","20");
```

```
      }
      ctrMkNode("fld",opt,-1,"/help/g_help",_("Опції
допомоги"),R_R___,"root",SARH_ID,3,"tp","str","cols","90","rows","10");
      return;
    }

    //> Процес управління на сторінці
    if(a_path == "/m_arch/per")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(messPeriod()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setMessPeriod(atoi(opt->text().c_str()));
    }
    else if(a_path == "/m_arch/size")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(messBufLen()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setMessBufLen(atoi(opt->text().c_str()));
    }
    else if(a_path == "/m_arch/view/tm")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))
      {
          opt->setText(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")));
          if( !atoi(opt->text().c_str()) )    opt-
>setText(TSYS::int2str(time(NULL)));
      }
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
          TBDS::genDBSet(nodePath()+"messTm",(atoi(opt-
>text().c_str())>=time(NULL))?"0":opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/size")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messSize","60",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messSize",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/cat")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messCat",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/lvl")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messLev","0",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messLev",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/view/archtor")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messArch","",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messArch",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/m_arch/lstAMess" && ctrChkNode(opt,"get",R_R___))
    {
      opt->childAdd("el")->setText("");
      opt->childAdd("el")->setText(BUF_ARCH_NM);
      vector<string> lsm, lsa;
      modList(lsm);
      for(unsigned i_m = 0; i_m < lsm.size(); i_m++)
```

```
      {
          at(lsm[i_m]).at().messList(lsa);
          for(unsigned i_a = 0; i_a < lsa.size(); i_a++)
            opt->childAdd("el")->setText(lsm[i_m]+"."+lsa[i_a]);
      }
    }
    else if(a_path == "/m_arch/view/mess" &&
ctrChkNode(opt,"get",R_R___,"root",SARH_ID))
    {
      vector<TMess::SRec> rec;
      time_t gtm = atoi(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")).c_str());
      if( !gtm ) gtm = time(NULL);
      int gsz = atoi(TBDS::genDBGet(nodePath()+"messSize","60",opt-
>attr("user")).c_str());
      messGet( gtm-gsz, gtm, rec,
             TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user")),
             atoi(TBDS::genDBGet(nodePath()+"messLev","0",opt-
>attr("user")).c_str()),
             TBDS::genDBGet(nodePath()+"messArch","",opt->attr("user")) );

      XMLNode *n_tm     = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0","",R_R___,"root",SARH_ID);
      XMLNode *n_tmu    = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/0a","",R_R___,"root",SARH_ID);
      XMLNode *n_cat    = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/1","",R_R___,"root",SARH_ID);
      XMLNode *n_lvl    = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/2","",R_R___,"root",SARH_ID);
      XMLNode *n_mess   = ctrMkNode("list",opt,-
1,"/m_arch/view/mess/3","",R_R___,"root",SARH_ID);
      for(int i_rec = rec.size()-1; i_rec >= 0; i_rec--)
      {
          if(n_tm)      n_tm->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].time));
          if(n_tmu)     n_tmu->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].utime));
          if(n_cat)     n_cat->childAdd("el")->setText(rec[i_rec].categ);
          if(n_lvl)     n_lvl->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].level));
          if(n_mess)    n_mess->childAdd("el")->setText(rec[i_rec].mess);
      }
    }
    else if(a_path == "/v_arch/per")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(valPeriod()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setValPeriod(atoi(opt->text().c_str()));
    }
    else if(a_path == "/v_arch/prior")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(valPrior()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))
      setValPrior(atoi(opt->text().c_str()));
    }
    else if(a_path == "/v_arch/nmb" && ctrChkNode(opt))
    {
      vector<string> list;
      valList(list);
      unsigned e_c = 0;
      for(unsigned i_a = 0; i_a < list.size(); i_a++)
          if(valAt(list[i_a]).at().startStat()) e_c++;
      opt->setText(TSYS::strMess(_("All: %d; Enabled: %d"),list.size(),e_c));
    }
    else if(a_path == "/br/va_" || a_path == "/v_arch/archs")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))
```

```
      {
          vector<string> list;
          valList(list);
          for(unsigned i_a=0; i_a < list.size(); i_a++)
            opt->childAdd("el")->setAttr("id",list[i_a])-
>setText(valAt(list[i_a]).at().name());
      }
      if(ctrChkNode(opt,"add",RWRWR_,"root",SARH_ID,SEC_WR))
      {
          string vid = TSYS::strEncode(opt->attr("id"),TSYS::oscdID);
          valAdd(vid); valAt(vid).at().setName(opt->text());
      }
      if(ctrChkNode(opt,"del",RWRWR_,"root",SARH_ID,SEC_WR))
      chldDel(mAval,opt->attr("id"),-1,1);
    }
    else if(a_path == "/help/g_help" &&
ctrChkNode(opt,"get",R_R___,"root",SARH_ID))    opt->setText(optDescr());
    else TSubSYS::cntrCmdProc(opt);
}

//************************************************
//* TTipArchivator                               *
//************************************************
TTipArchivator::TTipArchivator( const string &id ) : TModule(id)
{
    mVal = grpAdd("val_");
    mMess = grpAdd("mess_");
}

TTipArchivator::~TTipArchivator()
{
    nodeDelAll();
}

TArchiveS &TTipArchivator::owner()
{
    return (TArchiveS &)TModule::owner();
}

void TTipArchivator::messAdd(const string &name, const string &idb )
{
    chldAdd(mMess, AMess(name,idb));
}

void TTipArchivator::valAdd( const string &iid, const string &idb )
{
    chldAdd(mVal, AVal(iid,idb));
}

void TTipArchivator::cntrCmdProc( XMLNode *opt )
{
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TModule::cntrCmdProc(opt);
      ctrMkNode("area",opt,0,"/arch",_("Архіватори"));
      ctrMkNode("grp",opt,-1,"/br/mess_",_("Повідомлення
архіватора"),RWRWR_,"root",SARH_ID,2,"idm","1","idSz","20");
      ctrMkNode("grp",opt,-1,"/br/val_",_("Значення
архіватора"),RWRWR_,"root",SARH_ID,2,"idm","1","idSz","20");
      ctrMkNode("list",opt,-1,"/arch/mess",_("Повідомлення
архіваторів"),RWRWR_,"root",SARH_ID,5,"tp","br","idm","1","s_com","add,del","br_
pref","mess_","idSz","20");
      ctrMkNode("list",opt,-1,"/arch/val",_("Значення
архіваторів"),RWRWR_,"root",SARH_ID,5,"tp","br","idm","1","s_com","add,del","br_
pref","val_","idSz","20");
      return;
    }
    //> Процес управління на сторінці
```

```
    string a_path = opt->attr("path");
    if(a_path == "/br/mess_" || a_path == "/arch/mess")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))
      {
          vector<string> list;
          messList(list);
          for( unsigned i_a=0; i_a < list.size(); i_a++ )
            opt->childAdd("el")->setAttr("id",list[i_a])-
>setText(messAt(list[i_a]).at().name());
      }
      if(ctrChkNode(opt,"add",RWRWR_,"root",SARH_ID,SEC_WR))
      {
          string vid = TSYS::strEncode(opt->attr("id"),TSYS::oscdID);
          messAdd(vid); messAt(vid).at().setName(opt->text());
      }
      if(ctrChkNode(opt,"del",RWRWR_,"root",SARH_ID,SEC_WR))     messDel(opt-
>attr("id"),true);
    }
    else if(a_path == "/br/val_" || a_path == "/arch/val")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))
      {
          vector<string> list;
          valList(list);
          for(unsigned i_a=0; i_a < list.size(); i_a++)
            opt->childAdd("el")->setAttr("id",list[i_a])-
>setText(valAt(list[i_a]).at().name());
      }
      if(ctrChkNode(opt,"add",RWRWR_,"root",SARH_ID,SEC_WR))
      {
          string vid = TSYS::strEncode(opt->attr("id"),TSYS::oscdID);
          valAdd(vid); valAt(vid).at().setName(opt->text());
      }
      if(ctrChkNode(opt,"del",RWRWR_,"root",SARH_ID,SEC_WR))     valDel(opt-
>attr("id"),true);
    }
    else TModule::cntrCmdProc(opt);
}

//************************************************
//* Повідомлення архіватора                      *
//************************************************

//************************************************
//* TMArchivator                                 *
//************************************************
TMArchivator::TMArchivator(const string &iid, const string &idb, TElem *cf_el) :
    TConfig( cf_el ), run_st(false),
    m_id(cfg("ID").getSd()), m_name(cfg("NAME").getSd()),
m_dscr(cfg("DESCR").getSd()), m_addr(cfg("ADDR").getSd()),
    m_cat_o(cfg("CATEG").getSd()), m_start(cfg("START").getBd()),
m_level(cfg("LEVEL").getId()), m_db(idb)
{
    m_id = iid;
}

TCntrNode &TMArchivator::operator=( TCntrNode &node )
{
    TMArchivator *src_n = dynamic_cast<TMArchivator*>(&node);
    if( !src_n ) return *this;

    //> Конфігурація копіювання
    string tid = id();
    *(TConfig*)this = *(TConfig*)src_n;
    cfg("MODUL").setS(owner().modId());
    m_id = tid;
    m_db = src_n->m_db;
```

```
    if( src_n->startStat() && toStart() && !startStat() )
        start( );

    return *this;
}

void TMArchivator::postEnable( int flag )
{
    cfg("MODUL").setS(owner().modId());
}

void TMArchivator::preDisable( int flag )
{
    if( startStat() )   stop( );
}

void TMArchivator::postDisable(int flag)
{
    try
    {
      if( flag )
          SYS->db().at().dataDel(fullDB(),SYS-
>archive().at().nodePath()+tbl(),*this,true);
    }catch(TError err)
    { mess_warning(err.cat.c_str(),"%s",err.mess.c_str()); }
}

TTipArchivator &TMArchivator::owner( )    { return *(TTipArchivator*)nodePrev();
}

string TMArchivator::workId( )          { return owner().modId()+"."+id(); }

string TMArchivator::name()        { return (m_name.size())?m_name:m_id; }

string TMArchivator::tbl( )        { return
owner().owner().subId()+"_mess_proc"; }

void TMArchivator::load_( )
{
    if( !SYS->chkSelDB(DB()) ) return;
    SYS->db().at().dataGet(fullDB(),SYS->archive().at().nodePath()+tbl(),*this);
}

void TMArchivator::save_( )
{
    SYS->db().at().dataSet(fullDB(),SYS->archive().at().nodePath()+tbl(),*this);
}

void TMArchivator::categ( vector<string> &list )
{
    list.clear();
    string c_vl;
    for( int i_off = 0; (c_vl=TSYS::strSepParse(m_cat_o,0,';',&i_off)).size(); )
      list.push_back(c_vl);
}

bool TMArchivator::chkMessOK( const string &icateg, TMess::Type ilvl )
{
    vector<string> cat_ls;

    categ(cat_ls);

    if(ilvl >= level())
      for(unsigned i_cat = 0; i_cat < cat_ls.size(); i_cat++)
          if(TRegExp(cat_ls[i_cat], "p").test(icateg))
            return true;
   return false;
}
```

```
TVariant TMArchivator::objFuncCall( const string &iid, vector<TVariant> &prms,
const string &user )
{
    // bool status() - беремо статус архівування.
    if(iid == "status") return startStat();
    // int end() - беремо дані  архівування, час закінчення.
    if(iid == "end") return (int)end();
    // int begin() - беремо дані  архівування, час початку.
    if(iid == "begin") return (int)begin();

    //> Виклик функцій конфігурації
    TVariant cfRez = objFunc(iid, prms, user);
    if(!cfRez.isNull()) return cfRez;

    return TCntrNode::objFuncCall(iid, prms, user);
}

void TMArchivator::cntrCmdProc( XMLNode *opt )
{
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TCntrNode::cntrCmdProc(opt);
      ctrMkNode("SNMP_NMS_Application_Centric_Infrastructure_cntr",opt,-
1,"/",_("Повідомлення архіватора: ")+name(),RWRWR_,"root",SARH_ID);
      if(ctrMkNode("area",opt,-1,"/prm",_("Архіватор")))
      {
          if(ctrMkNode("area",opt,-1,"/prm/st",_("Стан")))
          {
            ctrMkNode("fld",opt,-
1,"/prm/st/st",_("Running"),RWRWR_,"root",SARH_ID,1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/prm/st/db",_("Архіватор
БД"),RWRWR_,"root","root",4,

"tp","str","dest","select","select","/db/list","help",TMess::labDB());
            ctrMkNode("fld",opt,-
1,"/prm/st/end",_("End"),R_R_R_,"root","root",1,"tp","time");
            ctrMkNode("fld",opt,-
1,"/prm/st/beg",_("Begin"),R_R_R_,"root","root",1,"tp","time");
          }
          if(ctrMkNode("area",opt,-1,"/prm/cfg",_("Конфігурація")))
          {
            ctrMkNode("fld",opt,-
1,"/prm/cfg/id",cfg("ID").fld().descr(),R_R_R_,"root",SARH_ID,1,"tp","str");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/nm",cfg("NAME").fld().descr(),RWRWR_,"root",SARH_ID,2,"tp","str","le
n","50");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/dscr",cfg("DESCR").fld().descr(),RWRWR_,"root",SARH_ID,3,"tp","str",
"cols","90","rows","3");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/addr",cfg("ADDR").fld().descr(),RWRWR_,"root",SARH_ID,1,"tp","str");
            ctrMkNode("fld",opt,-
1,"/prm/cfg/lvl",cfg("LEVEL").fld().descr(),RWRWR_,"root",SARH_ID,2,"tp","dec",
                "help",_("Отримуємо повідомлення для рівня більшого і
дорівнюючого цьому."));
            ctrMkNode("fld",opt,-
1,"/prm/cfg/cats",cfg("CATEG").fld().descr(),RWRWR_,"root",SARH_ID,2,"tp","str",
                "help",_("Шаблон категорії повідомлень або регулярне вираження у
процесі архівування, відокремлюються символом';'.\n"
                  "Використовуйте тимчасові символи для групового
виділення:\n  '*' - будь-які підрядки;\n  '?' - будь-які символи.\n"
                  "Регулярне вираження складається з символів'/'
(/mod_(System|LogicLev)/)."));
            ctrMkNode("fld",opt,-1,"/prm/cfg/start",_("To
start"),RWRWR_,"root",SARH_ID,1,"tp","bool");
          }
      }
```

```
      if(run_st && ctrMkNode("area",opt,-
1,"/mess",_("Messages"),R_R___,"root",SARH_ID))
      {
          ctrMkNode("fld",opt,-
1,"/mess/tm",_("Time"),RWRW__,"root",SARH_ID,1,"tp","time");
          ctrMkNode("fld",opt,-1,"/mess/size",_("Розмір
(s)"),RWRW__,"root",SARH_ID,1,"tp","dec");
          ctrMkNode("fld",opt,-1,"/mess/cat",_("Зразок
категорії"),RWRW__,"root",SARH_ID,2,"tp","str","help",
            _("Шаблон категорії повідомлень або регулярне вираження.\n"
                  "Використовуйте тимчасові символи для групового виділення:\n
'*' - будь-які підрядки;\n  '?' - будь-які символи.\n"
                "Регулярне вираження складається з символів'/'
(/mod_(System|LogicLev)/)."));
          ctrMkNode("fld",opt,-
1,"/mess/lvl",_("Level"),RWRW__,"root",SARH_ID,4,"tp","dec","min","0","max","7",
            "help",_("Отримуємо повідомлення для рівня більшого і дорівнюючого
цьому."));
          if(ctrMkNode("table",opt,-
1,"/mess/mess",_("Messages"),R_R___,"root",SARH_ID))
          {
            ctrMkNode("list",opt,-
1,"/mess/mess/0",_("Час"),R_R___,"root",SARH_ID,1,"tp","time");
            ctrMkNode("list",opt,-
1,"/mess/mess/0a",_("mcsec"),R_R___,"root",SARH_ID,1,"tp","dec");
            ctrMkNode("list",opt,-
1,"/mess/mess/1",_("Категорія"),R_R___,"root",SARH_ID,1,"tp","str");
            ctrMkNode("list",opt,-
1,"/mess/mess/2",_("Рівень"),R_R___,"root",SARH_ID,1,"tp","dec");
            ctrMkNode("list",opt,-
1,"/mess/mess/3",_("Повідомлення"),R_R___,"root",SARH_ID,1,"tp","str");
          }
      }
      return;
    }
    //> Процес управління на сторінці
    string a_path = opt->attr("path");
    if(a_path == "/prm/st/st")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
startStat() ? "1" : "0" );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     atoi(opt-
>text().c_str()) ? start() : stop();
    }
    else if(a_path == "/prm/st/db")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
DB() );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setDB( opt-
>text() );
    }
    else if(a_path == "/prm/st/end" && ctrChkNode(opt))          opt->setText(
TSYS::int2str(end()) );
    else if(a_path == "/prm/st/beg" && ctrChkNode(opt))          opt->setText(
TSYS::int2str(begin()) );
    else if(a_path == "/prm/cfg/id" && ctrChkNode(opt))          opt->setText(
id() );
    else if(a_path == "/prm/cfg/nm" )
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
name() );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setName( opt-
>text() );
    }
    else if(a_path == "/prm/cfg/dscr")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
dscr() );
```

```
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setDscr( opt-
>text() );
    }
    else if(a_path == "/prm/cfg/addr")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
addr() );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setAddr( opt-
>text() );
    }
    else if(a_path == "/prm/cfg/lvl")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(TSYS::int2str(level()));
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setLevel(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/prm/cfg/start")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt->setText(
toStart() ? "1" : "0" );
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     setToStart(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/prm/cfg/cats")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SARH_ID,SEC_RD))     opt-
>setText(m_cat_o);
      if(ctrChkNode(opt,"set",RWRWR_,"root",SARH_ID,SEC_WR))     { m_cat_o =
opt->text(); modif(); }
    }
    else if(a_path == "/mess/tm")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))
      {
          opt->setText(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")));
          if(!atoi(opt->text().c_str()))  opt-
>setText(TSYS::int2str(time(NULL)));
      }
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
          TBDS::genDBSet(nodePath()+"messTm",(atoi(opt-
>text().c_str())>=time(NULL))?"0":opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/size")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messSize","10",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messSize",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/cat")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messCat",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/lvl")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root",SARH_ID,SEC_RD))     opt-
>setText(TBDS::genDBGet(nodePath()+"messLev","0",opt->attr("user")));
      if(ctrChkNode(opt,"set",RWRW__,"root",SARH_ID,SEC_WR))
      TBDS::genDBSet(nodePath()+"messLev",opt->text(),opt->attr("user"));
    }
    else if(a_path == "/mess/mess" && run_st &&
ctrChkNode(opt,"get",R_R___,"root",SARH_ID))
    {
      vector<TMess::SRec> rec;
```

```
      time_t end = atoi(TBDS::genDBGet(nodePath()+"messTm","0",opt-
>attr("user")).c_str());
      if( !end ) end = time(NULL);
      time_t beg = end - atoi(TBDS::genDBGet(nodePath()+"messSize","10",opt-
>attr("user")).c_str());
      string cat = TBDS::genDBGet(nodePath()+"messCat","",opt->attr("user"));
      char   lev = atoi(TBDS::genDBGet(nodePath()+"messLev","0",opt-
>attr("user")).c_str());

      get( beg, end, rec, cat, lev );

      XMLNode *n_tm     = ctrMkNode("list",opt,-
1,"/mess/mess/0","",R_R___,"root",SARH_ID);
      XMLNode *n_tmu    = ctrMkNode("list",opt,-
1,"/mess/mess/0a","",R_R___,"root",SARH_ID);
      XMLNode *n_cat    = ctrMkNode("list",opt,-
1,"/mess/mess/1","",R_R___,"root",SARH_ID);
      XMLNode *n_lvl    = ctrMkNode("list",opt,-
1,"/mess/mess/2","",R_R___,"root",SARH_ID);
      XMLNode *n_mess   = ctrMkNode("list",opt,-
1,"/mess/mess/3","",R_R___,"root",SARH_ID);
      for(int i_rec = rec.size()-1; i_rec >= 0; i_rec--)
      {
          if(n_tm)      n_tm->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].time));
          if(n_tmu)     n_tmu->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].utime));
          if(n_cat)     n_cat->childAdd("el")->setText(rec[i_rec].categ);
          if(n_lvl)     n_lvl->childAdd("el")-
>setText(TSYS::int2str(rec[i_rec].level));
          if(n_mess)    n_mess->childAdd("el")->setText(rec[i_rec].mess);
      }
    }
    else TCntrNode::cntrCmdProc(opt);
}
```

tmodule.cpp – робота з модулями

```
//WebSNMP_NMS_Application_Centric_Infrastructure системний файл: tmodule.cpp

#include <sys/types.h>
#include <sys/stat.h>
#include <stdarg.h>
#include <unistd.h>
#include <dlfcn.h>
#include <string.h>
#include <libintl.h>

#include "tsys.h"
#include "terror.h"
#include "tmess.h"
#include "tsubsys.h"
#include "tmodule.h"

using namespace SNMP_NMS_Application_Centric_Infrastructure;

//************************************************
//* TModule                                      *
//************************************************
const char *TModule::l_info[] =
    {"Модуль","Ім'я","Тип","Джерело","Версія","Автор","Дескриптор","Ліцензія"};

TModule::TModule( const string &id ) : mId(id)
{
    lc_id = string("oscd_")+mId;
    bindtextdomain(lc_id.c_str(),LOCALEDIR);

    //> Переведення динамічного рядка
#if 0
    char mess[][100] = { _("Автор"), _("Ліцензія") };
#endif
}

TModule::~TModule(  )
{
    //> Очищення списку експортуємих функцій
    for(unsigned i = 0; i < m_efunc.size(); i++)
        delete m_efunc[i];
}

string TModule::modName()
{
    return mName;
}

void TModule::postEnable( int flag )
{
    if(flag&TCntrNode::NodeRestore) return;

    mess_info(nodePath().c_str(),_("З'єднання з модулем!"));
}

TSubSYS &TModule::owner( )    { return *(TSubSYS*)nodePrev(); }

void TModule::modFuncList( vector<string> &list )
{
    list.clear();
    for(unsigned i = 0; i < m_efunc.size(); i++)
      list.push_back(m_efunc[i]->prot);
}

bool TModule::modFuncPresent( const string &prot )
{
    for(unsigned i = 0; i < m_efunc.size(); i++)
```

```
      if(m_efunc[i]->prot == prot)
          return true;
    return false;
}

TModule::ExpFunc &TModule::modFunc( const string &prot )
{
    for(unsigned i = 0; i < m_efunc.size(); i++)
      if(m_efunc[i]->prot == prot) return *m_efunc[i];
    throw TError(nodePath().c_str(),_("Функція '%s' не представлена у
модулі!"),prot.c_str());
}

void TModule::modFunc( const string &prot, void (TModule::**offptr)() )
{
    *offptr = modFunc(prot).ptr;
}

void TModule::modInfo( vector<string> &list )
{
    for(unsigned i_opt = 0; i_opt < sizeof(l_info)/sizeof(char *); i_opt++)
      list.push_back(l_info[i_opt]);
}

string TModule::modInfo( const string &name )
{
    string info;

    if(name == l_info[0])     info = mId;
    else if(name == l_info[1]) info = mName;
    else if(name == l_info[2]) info = mType;
    else if(name == l_info[3]) info = mSource;
    else if(name == l_info[4]) info = mVers;
    else if(name == l_info[5]) info = mAuthor;
    else if(name == l_info[6]) info = mDescr;
    else if(name == l_info[7]) info = mLicense;

    return info;
}

void TModule::cntrCmdProc( XMLNode *opt )
{
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TCntrNode::cntrCmdProc(opt);
      ctrMkNode("SNMP_NMS_Application_Centric_Infrastructure_cntr",opt,-
1,"/",_("Module: ")+modId(),R_R_R_);
      ctrMkNode("branches",opt,-1,"/br","",R_R_R_);
      if(TUIS::icoPresent(owner().subId()+"."+modId())) ctrMkNode("img",opt,-
1,"/ico","",R_R_R_);
      if(ctrMkNode("area",opt,-1,"/help",_("Help")))
          if(ctrMkNode("area",opt,-1,"/help/m_inf",_("Модуль інформації")))
          {
            vector<string> list;
            modInfo(list);
            for(unsigned i_l = 0; i_l < list.size(); i_l++)
                ctrMkNode("fld",opt,-
1,(string("/help/m_inf/")+list[i_l]).c_str(),_(list[i_l].c_str()),R_R_R_,"root",
"root",1,"tp","str");
          }
      return;
    }

    //> Процес управління на сторінці
    string a_path = opt->attr("path");
    if(a_path == "/ico" && ctrChkNode(opt))
    {
      string itp;
```

```
      opt-
>setText(TSYS::strEncode(TUIS::icoGet(owner().subId()+"."+modId(),&itp),TSYS::ba
se64));
      opt->setAttr("tp",itp);
    }
    else if(a_path.substr(0,11) == "/help/m_inf" && ctrChkNode(opt))
      opt->setText(modInfo(TSYS::pathLev(a_path,2)));
    else TCntrNode::cntrCmdProc(opt);
}

const char *TModule::I18N( const char *mess )
{
    const char *rez = Mess->I18N(mess,lc_id.c_str());
    if( !strcmp(mess,rez) ) rez = _(mess);
    return rez;
}
```

tparamcontr.cpp – параметри управління системою моніторингу продуктивності мережі на базі SNMP/NMS

```
//WebSNMP_NMS_Application_Centric_Infrastructure системний файл: tparamcontr.cpp

#include "tbds.h"
#include "tsys.h"
#include "tmess.h"
#include "tdaqs.h"
#include "tcontroller.h"
#include "ttipdaq.h"
#include "ttiparam.h"
#include "tparamcontr.h"

using namespace SNMP_NMS_Application_Centric_Infrastructure;

//************************************************
//* TParamContr                                  *
//************************************************
TParamContr::TParamContr( const string &name, TTipParam *tpprm ) :
TConfig(tpprm), m_en(false), tipparm(tpprm)
{
    setId(name);
    setName(name);
}

TParamContr::~TParamContr( )
{
    nodeDelAll();
}

TCntrNode &TParamContr::operator=( TCntrNode &node )
{
    TParamContr *src_n = dynamic_cast<TParamContr*>(&node);
    if(!src_n) return *this;

    //> Перевіряємо тип параметрів й змінюємо їх, якщо вони змінні, або нижчі
    if(type().name != src_n->type().name && owner().owner().tpPrmToId(src_n-
>type().name) >= 0)
    {
      if(enableStat()) disable();
      setType(src_n->type().name);
    }

    //> Конфігурація копіювання
    string tid = id();
    *(TConfig*)this = *(TConfig*)src_n;
    setId(tid);

    //> Дозволяємо нові параметри
    if(src_n->enableStat() && toEnable( ) && !enableStat())enable();

    return *this;
}

TController &TParamContr::owner( ) { return *(TController*)nodePrev(); }

string TParamContr::name( )   { string nm = cfg("NAME").getS(); return nm.size()
? nm : id(); }

void TParamContr::setName( const string &inm ) { cfg("NAME").setS(inm); }

string TParamContr::descr( ) { return cfg("DESCR").getS(); }

void TParamContr::setDescr( const string &idsc ){ cfg("DESCR").setS(idsc); }

void TParamContr::postEnable(int flag)
{
```

```
    TValue::postEnable(flag);

    if(!vlCfg())  setVlCfg(this);
    if(!vlElemPresent(&SYS->daq().at().errE()))
      vlElemAtt(&SYS->daq().at().errE());
}

void TParamContr::preDisable(int flag)
{
    //> Видаляємо або зупиняємо архівування
    vector<string> a_ls;
    vlList(a_ls);

    for(unsigned i_a = 0; i_a < a_ls.size(); i_a++)
      if(!vlAt(a_ls[i_a]).at().arch().freeStat())
      {
          string arh_id = vlAt(a_ls[i_a]).at().arch().at().id();
          if(flag) SYS->archive().at().valDel(arh_id,true);
          else SYS->archive().at().valAt(arh_id).at().stop();
      }

    if(enableStat())    disable();
}

void TParamContr::postDisable(int flag)
{
    if(flag)
    {
      //> Видаляємо параметри з БД
      try
      {
          SYS->db().at().dataDel(owner().DB()+"."+owner().cfg(type().db).getS(),

owner().owner().nodePath()+owner().cfg(type().db).getS(),*this,true);
      }catch(TError err) { mess_err(err.cat.c_str(),"%s",err.mess.c_str()); }
    }
}

void TParamContr::load_( )
{
    if(!SYS->chkSelDB(owner().DB())) return;

    cfgViewAll(true);
    SYS->db().at().dataGet(owner().DB()+"."+owner().cfg(type().db).getS(),

owner().owner().nodePath()+owner().cfg(type().db).getS(),*this);
}

void TParamContr::save_( )
{
    SYS->db().at().dataSet( owner().DB()+"."+owner().cfg(type().db).getS(),

owner().owner().nodePath()+owner().cfg(type().db).getS(),*this );

    //> Зберігаємо архіви
    vector<string> a_ls;
    vlList(a_ls);
    for(unsigned i_a = 0; i_a < a_ls.size(); i_a++)
      if(!vlAt(a_ls[i_a]).at().arch().freeStat())
          vlAt(a_ls[i_a]).at().arch().at().save();
}

bool TParamContr::cfgChange( TCfg &cfg )  { modif( ); return true; }

TParamContr & TParamContr::operator=( TParamContr & PrmCntr )
{
    TConfig::operator=(PrmCntr);

    return *this;
```

```
}

void TParamContr::enable()
{
    m_en = true;
}

void TParamContr::disable()
{
    m_en = false;
}

void TParamContr::vlGet( TVal &val )
{
    if( val.name() == "err" )
    {
      if( !enableStat() ) val.setS(_("1:Параметри заборонені."),0,true);
      else if( !owner().startStat( ) ) val.setS(_("2:Контролер
зупинено."),0,true);
      else val.setS("0",0,true);
    }
}

void TParamContr::setId( const string &vl )
{
    cfg("SHIFR").setS(vl);
}

void TParamContr::setType( const string &tpId )
{
    if(enableStat() || tpId == type().name ||
!owner().owner().tpPrmPresent(tpId))     return;

    setNodeMode(TCntrNode::Disable);

    try
    {
      //> Чекаємо поки роз'єднаються інші
      while(nodeUse(true) > 1) usleep(1000);
      //> Видаляємо з БД
      postDisable(true);

      //> Створюємо тимчасову структуру
      TConfig tCfg(&type());
      tCfg = *(TConfig*)this;

      //> Встановлюємо нову конфігурацію структури
      tipparm = &owner().owner().tpPrmAt(owner().owner().tpPrmToId(tpId));
      setElem(tipparm);

      //> Відновлюємо конфігурацію
      *(TConfig*)this = tCfg;
    }catch(...) { }
    setNodeMode(TCntrNode::Enable);
    setVlCfg(this);
    modif();
}
TVariant TParamContr::objFuncCall( const string &iid, vector<TVariant> &prms,
const string &user )
{
    //> Виклик функцій конфігурації
    TVariant cfRez = objFunc(iid, prms, user);
    if(!cfRez.isNull()) return cfRez;
    return TValue::objFuncCall(iid, prms, user);
}
void TParamContr::cntrCmdProc( XMLNode *opt )
{
    string a_path = opt->attr("path");
    //> Сервіс управління процесами
```

```
    if(a_path.substr(0,6) == "/serv/")    { TValue::cntrCmdProc(opt); return; }
    //> Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TValue::cntrCmdProc(opt);
      ctrMkNode("SNMP_NMS_Application_Centric_Infrastructure_cntr",opt,-
1,"/",_("Параметр: ")+name(),RWRWR_,"root",SDAQ_ID);
      if(ctrMkNode("area",opt,0,"/prm",_("Parameter")))
      {
          if(ctrMkNode("area",opt,-1,"/prm/st",_("Стан")))
          {
            if(!enableStat() && owner().owner().tpPrmSize() > 1)
                ctrMkNode("fld",opt,-
1,"/prm/st/type",_("Тип"),RWRWR_,"root",SDAQ_ID,4,"tp","str","dest","select","se
lect","/prm/tpLst",
                  "help",_("Змінюємо тип керівництва до останіх даних для
специфічних конфігурацій."));
            else ctrMkNode("fld",opt,-
1,"/prm/st/type",_("Тип"),R_R_R_,"root",SDAQ_ID,1,"tp","str");
            if(owner().enableStat())
                ctrMkNode("fld",opt,-
1,"/prm/st/en",_("Дозволено"),RWRWR_,"root",SDAQ_ID,1,"tp","bool");
          }
          if(ctrMkNode("area",opt,-1,"/prm/cfg",_("Configuration")))
            TConfig::cntrCmdMake(opt,"/prm/cfg",0,"root",SDAQ_ID,RWRWR_);
      }
        return;
    }
    //> Процес управління на сторінці
    if(a_path == "/prm/st/type")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SDAQ_ID,SEC_RD))     opt-
>setText(type().name);
      if(ctrChkNode(opt,"set",RWRWR_,"root",SDAQ_ID,SEC_WR))     setType(opt-
>text());
    }
    else if(a_path == "/prm/st/en")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root",SDAQ_ID,SEC_RD))     opt-
>setText(enableStat()?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root",SDAQ_ID,SEC_WR))
      {
          if(!owner().enableStat()) throw TError(nodePath().c_str(),_("Контролер
не запустився!"));
          else atoi(opt->text().c_str())?enable():disable();
      }
    }
    else if(a_path.substr(0,8) == "/prm/cfg")
TConfig::cntrCmdProc(opt,TSYS::pathLev(a_path,2),"root",SDAQ_ID,RWRWR_);
    else if(a_path == "/prm/tmplList" && ctrChkNode(opt))
    {
      vector<string> lls, ls;
      SYS->daq().at().tmplLibList(lls);
      for(unsigned i_l = 0; i_l < lls.size(); i_l++)
      {
          SYS->daq().at().tmplLibAt(lls[i_l]).at().list(ls);
          for(unsigned i_t = 0; i_t < ls.size(); i_t++)
            opt->childAdd("el")->setText(lls[i_l]+"."+ls[i_t]);
      }
    }
    else if(a_path == "/prm/tpLst" && ctrChkNode(opt))
      for(unsigned i_tp = 0; i_tp < owner().owner().tpPrmSize(); i_tp++)
          opt->childAdd("el")->setAttr("id",owner().owner().tpPrmAt(i_tp).name)-
>setText(owner().owner().tpPrmAt(i_tp).descr);
    else TValue::cntrCmdProc(opt);
}
```

main.cpp – головна програма

```
//Файл системи моніторингу продуктивності мережі на базі SNMP/NMS Centric
Infrastructure: main.cpp

#include <getopt.h>
#include "terror.h"
#include "tmess.h"
#include "tsys.h"

using namespace SNMP_NMS_Application_Centric_Infrastructure;

int main(int argc, char *argv[], char *envp[] )
{
    int rez = 0;

    //Перевірка початку роботи режиму основного процесу
    int next_opt;
    optind=opterr=0;
    struct option long_opt[] = { {"Режим основного процесу" ,0,NULL,'d'}, {NULL
,0,NULL,0  } };
    while((next_opt=getopt_long(argc,argv,"",long_opt,NULL)) != -1)
      if( next_opt == 'd' )
      {
          printf("Початок роботи режиму основного процесу!\n");
          int pid = fork();
          if( pid == -1 )
          {
            printf("Помилка: неможливо створити новий процес!\n");
            return -1;
          }
          if( pid != 0 )      return 0;

          //Готується оточення режиму основного процесу
          setsid();
          break;
      }

    try
    {
      SYS = new TSYS(argc,argv,envp);

      SYS->load();
      if( (rez=SYS->stopSignal()) > 0 ) return rez;
      rez = SYS->start();

      delete SYS;
    }catch(TError err) { mess_err(err.cat.c_str(),"%s",err.mess.c_str()); }

    printf("Система моніторингу продуктивності мережі на базі SNMP/NMS Centric
Infrastructure коректно працює з даними %d.\n",rez);

    return rez;
}
```

```
//WebSNMP_NMS_Application_Centric_Infrastructure системний файл: tsys.cpp

#include <features.h>
#include <ieee754.h>
#include <syscall.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/utsname.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <getopt.h>
#include <stdio.h>
#include <signal.h>
#include <stdarg.h>
#include <stdlib.h>
#include <langinfo.h>
#include <zlib.h>

#include "terror.h"
#include "tmess.h"
#include "tsys.h"

using namespace SNMP_NMS_Application_Centric_Infrastructure;

//Поточна зміна доступу
TMess *SNMP_NMS_Application_Centric_Infrastructure::Mess;
TSYS  *SNMP_NMS_Application_Centric_Infrastructure::SYS;
bool TSYS::finalKill = false;
pthread_key_t TSYS::sTaskKey;

TSYS::TSYS( int argi, char ** argb, char **env ) : argc(argi), argv((const char
**)argb), envp((const char **)env),
    mUser("root"),
mConfFile("/etc/SNMP_NMS_Application_Centric_Infrastructure.xml"),
mId("EmptySt"), mName(_("Порожня Станція")), mIcoDir("./icons/"), mModDir("./"),
    mWorkDB("<cfg>"), mSaveAtExit(false), mSavePeriod(0), rootModifCnt(0),
mStopSignal(-1), mMultCPU(false)
{
    finalKill = false;
    SYS = this;         //Ініціалізуємо значення глобальних зміних доступу
    mSubst = grpAdd("sub_",true);
    nodeEn();
    pthread_key_create(&sTaskKey, NULL);

    Mess = new TMess();

    if(getenv("USER")) mUser = getenv("USER");

    //> Ініціалізуємо системний годинник
    clkCalc();

#if __GLIBC_PREREQ(2,4)
    //> Multi CPU дозволяють перевірку
    cpu_set_t cpuset;
    CPU_ZERO(&cpuset);
    CPU_SET(1,&cpuset);
    mMultCPU = !pthread_setaffinity_np(pthread_self(), sizeof(cpu_set_t),
&cpuset);
#endif

    //> Встановлюємо сигнальні програми обробки
```

```
    signal(SIGINT,sighandler);
    signal(SIGTERM,sighandler);
    //signal(SIGCHLD,sighandler);
    signal(SIGALRM,sighandler);
    signal(SIGPIPE,sighandler);
    //signal(SIGFPE,sighandler);
    //signal(SIGSEGV,sighandler);
    signal(SIGABRT,sighandler);
}

TSYS::~TSYS( )
{
    finalKill = true;

    //Видаляємо всі вузли в команді управління
    del("ModSched");
    del("UI");
    del("Special");
    del("Archive");
    del("DAQ");
    del("Protocol");
    del("Transport");
    del("Security");
    del("BD");

    delete Mess;
    pthread_key_delete(sTaskKey);
}

string TSYS::host( )
{
    utsname ubuf; uname(&ubuf);
    return ubuf.nodename;
}

string TSYS::workDir( )
{
    char buf[STR_BUF_LEN];
    return getcwd(buf,sizeof(buf));
}

void TSYS::setWorkDir( const string &wdir )
{
    if(workDir() == wdir) return;
    if(chdir(wdir.c_str()) != 0)
      mess_warning(nodePath().c_str(),_("Змініть робочу директорію'%s' Помилка:
%s. Можливо поточний каталог вже встановлено правильно'%s'."),
          wdir.c_str(),strerror(errno),workDir().c_str());
    modif( );
}

XMLNode *TSYS::cfgNode( const string &path, bool create )
{
    string s_el, ndNm;

    XMLNode *t_node = &rootN;
    if(t_node->name() != "WebSNMP_NMS_Application_Centric_Infrastructure")
    {
      if(!create) return NULL;
      t_node->setName("WebSNMP_NMS_Application_Centric_Infrastructure");
    }

    for(int l_off = 0, nLev = 0; true; nLev++)
    {
        s_el = TSYS::pathLev(path,0,true,&l_off);
        if(s_el.empty()) return t_node;
        bool ok = false;
        for(unsigned i_f = 0; !ok && i_f < t_node->childSize(); i_f++)
            if(t_node->childGet(i_f)->attr("id") == s_el)
```

```
            {
                t_node = t_node->childGet(i_f);
                ok = true;
            }
        if(!ok)
        {
          if(!create)   return NULL;
          ndNm = "prm";
          switch(nLev)
          {
            case 0: ndNm = "station";     break;
            case 1: if(s_el.compare(0,4,"sub_") == 0) ndNm = "node";   break;
            case 2: if(s_el.compare(0,4,"mod_") == 0) ndNm = "node";   break;
          }
          if(ndNm == "prm") t_node = t_node->childIns(0,ndNm)-
>setAttr("id",s_el);
          else t_node = t_node->childAdd(ndNm)->setAttr("id",s_el);
        }
    }
    return t_node;
}

string TSYS::int2str( int val, TSYS::IntView view )
{
    char buf[STR_BUF_LEN];
    if(view == TSYS::Dec)     snprintf(buf,sizeof(buf),"%d",val);
    else if(view == TSYS::Oct)      snprintf(buf,sizeof(buf),"%o",val);
    else if(view == TSYS::Hex)      snprintf(buf,sizeof(buf),"%x",val);

    return buf;
}

string TSYS::uint2str( unsigned val, IntView view )
{
    char buf[STR_BUF_LEN];
    if(view == TSYS::Dec)     snprintf(buf,sizeof(buf),"%u",val);
    else if(view == TSYS::Oct)      snprintf(buf,sizeof(buf),"%o",val);
    else if(view == TSYS::Hex)      snprintf(buf,sizeof(buf),"%x",val);

    return buf;
}

string TSYS::ll2str( int64_t val, IntView view )
{
    char buf[STR_BUF_LEN];
    if(view == TSYS::Dec)     snprintf(buf,sizeof(buf),"%lld",val);
    else if(view == TSYS::Oct)      snprintf(buf,sizeof(buf),"%llo",val);
    else if(view == TSYS::Hex)      snprintf(buf,sizeof(buf),"%llx",val);

    return buf;
}

string TSYS::real2str( double val, int prec, char tp )
{
    char buf[STR_BUF_LEN];
    if(tp == 'g') snprintf(buf,sizeof(buf),"%.*g",prec,val);
    else if(tp == 'e') snprintf(buf,sizeof(buf),"%.*e",prec,val);
    else snprintf(buf,sizeof(buf),"%.*f",prec,val);

    return buf;
}

string TSYS::time2str( time_t itm, const string &format )
{
    struct tm tm_tm;
    localtime_r(&itm,&tm_tm);
    char buf[100];
    int ret = strftime(buf, sizeof(buf), format.empty()?"%d-%m-%Y
%H:%M:%S":format.c_str(), &tm_tm);
```

```
    return (ret > 0) ? string(buf,ret) : string("");
}

string TSYS::time2str( double utm )
{
    if(utm < 1e-6) return "0";
    int lev = 0;
    int days = (int)floor(utm/(24*60*60*1e6));
    int часи = (int)floor(utm/(60*60*1e6))%24;
    int mins = (int)floor(utm/(60*1e6))%60;
    double usec = utm - 1e6*(days*24*60*60 + часи*60*60 + mins*60);

    string rez;
    if(days)            { rez += TSYS::int2str(days)+_("day"); lev =
vmax(lev,6); }
    if(часи)            { rez += (rez.size()?"
":"")+TSYS::int2str(часи)+_("годин"); lev = vmax(lev,5); }
    if(mins && lev < 6) { rez += (rez.size()?"
":"")+TSYS::int2str(mins)+_("хвилин"); lev = vmax(lev,4); }
    if((1e-6*usec) > 0.5 && lev < 5)      { rez += (rez.size()?"
":"")+TSYS::real2str(1e-6*usec,3)+_("секунд"); lev = vmax(lev,3); }
    else if((1e-3*usec) > 0.5 && !lev)    { rez += (rez.size()?"
":"")+TSYS::real2str(1e-3*usec,4)+_("мікросекунд"); lev = vmax(lev,2); }
    else if(usec > 0.5 && !lev)           { rez += (rez.size()?"
":"")+TSYS::real2str(usec,4)+_("us"); lev = vmax(lev,1); }
    else if(!lev) rez += (rez.size()?" ":"")+TSYS::real2str(1e3*usec,4)+_("ns");
    return rez;
}

string TSYS::cpct2str( double cnt )
{
    if(cnt > 0.2*pow(2,80))   return
TSYS::real2str(cnt/pow(2,80),3,'g')+_("YiB");
    if(cnt > 0.2*pow(2,70))   return
TSYS::real2str(cnt/pow(2,70),3,'g')+_("ZiB");
    if(cnt > 0.2*pow(2,60))   return
TSYS::real2str(cnt/pow(2,60),3,'g')+_("EiB");
    if(cnt > 0.2*pow(2,50))   return
TSYS::real2str(cnt/pow(2,50),3,'g')+_("PiB");
    if(cnt > 0.2*pow(2,40))   return
TSYS::real2str(cnt/pow(2,40),3,'g')+_("TiB");
    if(cnt > 0.2*pow(2,30))   return
TSYS::real2str(cnt/pow(2,30),3,'g')+_("GiB");
    if(cnt > 0.2*pow(2,20))   return
TSYS::real2str(cnt/pow(2,20),3,'g')+_("MiB");
    if(cnt > 0.2*pow(2,10))   return
TSYS::real2str(cnt/pow(2,10),3,'g')+_("KiB");
    return TSYS::real2str(cnt,3,'g')+_("B");
}

string TSYS::addr2str( void *addr )
{
    char buf[sizeof(void*)*2+3];
    snprintf(buf,sizeof(buf),"%p",addr);

    return buf;
}

void *TSYS::str2addr( const string &str )
{
    return (void *)strtoul(str.c_str(),NULL,16);
}

string TSYS::strNoSpace( const string &val )
{
    int beg = -1, end = -1;

    for(unsigned i_s = 0; i_s < val.size(); i_s++)
      if(val[i_s] != ' ' && val[i_s] != '\n' && val[i_s] != '\t')
```

```
      {
          if(beg < 0) beg = i_s;
          end = i_s;
      }

    return (beg>=0) ? val.substr(beg,end-beg+1) : "";
}

string TSYS::strMess( const char *fmt, ... )
{
    char str[STR_BUF_LEN];
    va_list argptr;

    va_start(argptr,fmt);
    vsnprintf(str,sizeof(str),fmt,argptr);
    va_end(argptr);

    return str;
}

string TSYS::optDescr( )
{
    utsname buf;
    uname(&buf);
    return TSYS::strMess(_(
      "**********************************************************************
**\n"
      "********** %s v%s (%s-%s). *********\n"
      "**********************************************************************
**\n\n"
      "======================================================================
==\n"
      "========================= Загальносистемні опції
======================\n"
      "======================================================================
==\n"
      "-h, --help             Інформаційні повідомлення про опції системи
моніторингу продуктивності мережі на базі SNMP/NMS.\n"
      "    --Config=<path>    Шлях до конфігураційного файлу.\n"
      "    --Station=<id>     Ідентифікатор станції.\n"
      "    --режим основного процесу           Початок роботи режиму основного
процесу.\n"
      "    --MessLev=<level>  Повідомлення процесів<level> (0-7).\n"
      "    --log=<direct>     Повідомлення про :\n"
      "                         <direct> & 1 - події у системі;\n"
      "                         <direct> & 2 - ввод/вивід у системі;\n"
      "                         <direct> & 4 - помилки у системі;\n"
      "                         <direct> & 8 - архівація у системі;\n"
      "---------- Конфігураційний файл параметрів робочої станції'%s' ---------
--\n"
      "StName     <nm>  Ім’я робочої станції.\n"
      "WorkDB     <Type.Name> Робоча база даних (тип та ім’я).\n"
      "Workdir    <path>      Робоча директорія.\n"
      "IcoDir     <path>      Директорія іконок.\n"
      "ModDir     <path>      Директорія модулів.\n"
      "MessLev    <level>     Повідомлення <рівні> (0-7).\n"
      "LogTarget  <direction> Повідомлення про :\n"
      "                           <direct> & 1 - події у системі;\n"
      "                           <direct> & 2 - ввод/вивід у системі;\n"
      "                           <direct> & 4 - помилки у системі;\n"
      "                           <direct> & 8 - архівація у системі;\n"
      "Lang2CodeBase <lang>   Базова мова для переведення тексту, двосимвольний
код .\n"
      "SaveAtExit <true>      Збереження змін у системі перед виходом.\n"
      "SavePeriod <sec> Збереження періоду роботи  у системі .\n\n"),
      PACKAGE_NAME,VERSION,buf.sysname,buf.release,nodePath().c_str());
}

bool TSYS::cfgFileLoad( )
```

```
{
    bool cmd_help = false;

    //================ Редагування параметрів=========================
    int next_opt;
    const char *short_opt="h";
    struct option long_opt[] =
    {
      {"help"     ,0,NULL,'h'},
      {"Config"   ,1,NULL,'f'},
      {"Station"  ,1,NULL,'s'},
      {NULL       ,0,NULL,0  }
    };

    optind=opterr=0;
    do
    {
      next_opt=getopt_long(argc,(char * const *)argv,short_opt,long_opt,NULL);
      switch(next_opt)
      {
          case 'h':
            fprintf(stdout,"%s",optDescr().c_str());
            Mess->setMessLevel(7);
            cmd_help = true;
            break;
          case 'f': mConfFile = optarg; break;
          case 's': mId = optarg; break;
          case -1 : break;
      }
    } while(next_opt != -1);

    //Завантажуємо кофігураційний файл
    int hd = open(mConfFile.c_str(),O_RDONLY);
    if(hd < 0) mess_err(nodePath().c_str(),_("Кофігураційний файл '%s' Помилка:
%s"),mConfFile.c_str(),strerror(errno));
    else
    {
      string s_buf;
      int cf_sz = lseek(hd,0,SEEK_END);
      if(cf_sz > 0)
      {
          lseek(hd,0,SEEK_SET);
          char *buf = (char *)malloc(cf_sz+1);
          read(hd,buf,cf_sz);
          buf[cf_sz] = 0;
          s_buf = buf;
          free(buf);
      }
      close(hd);

      try
      {
          ResAlloc res(nodeRes(),true);
          rootN.load(s_buf,true);
          if(rootN.name() == "WebSNMP_NMS_Application_Centric_Infrastructure")
          {
            XMLNode *stat_n = NULL;
            for(int i_st = rootN.childSize()-1; i_st >= 0; i_st--)
                if(rootN.childGet(i_st)->name() == "station")
                {
                  stat_n = rootN.childGet(i_st);
                  if(stat_n->attr("id") == mId) break;
                }
            if(stat_n && stat_n->attr("id") != mId)
            {
                mess_warning(nodePath().c_str(),_("Робоча станція '%s' не
представлена у  кофігураційному файлі Використайте '%s' конфігурацію робочої
станції!"),
                  mId.c_str(), stat_n->attr("id").c_str());
```

```
                mId     = stat_n->attr("id");
              }
              if(!stat_n) rootN.clear();
            } else rootN.clear();
            if(!rootN.childSize()) mess_err(nodePath().c_str(),_("Помилка
конфігурації '%s'!"),mConfFile.c_str());
            rootModifCnt = 0;
        }
        catch(TError err) { mess_err(nodePath().c_str(),_("Завантажуємо
кофігураційний файл Помилка: %s"),err.mess.c_str() ); }
    }

    return cmd_help;
}

void TSYS::cfgFileSave( )
{
    ResAlloc res(nodeRes(),true);
    if(!rootModifCnt) return;
    int hd = open(mConfFile.c_str(), O_CREAT|O_TRUNC|O_WRONLY, 0664);
    if(hd < 0) mess_err(nodePath().c_str(),_("Кофігураційний файл '%s' Помилка:
%s"),mConfFile.c_str(),strerror(errno));

    string rezFile = rootN.save(XMLNode::XMLHeader);
    int rez = write(hd, rezFile.data(), rezFile.size());
    if(rez != (int)rezFile.size()) mess_err(nodePath().c_str(),_("Помилка запису
конфігурації. %s"),mConfFile.c_str(),((rez<0)?strerror(errno):""));
    rootModifCnt = 0;
    rootFlTm = time(NULL);
}

void TSYS::cfgPrmLoad( )
{
    //Системні параметри
    mName =
TBDS::genDBGet(nodePath()+"StName",name(),"root",TBDS::UseTranslate);
    mWorkDB = TBDS::genDBGet(nodePath()+"WorkDB",workDB(),"root",TBDS::OnlyCfg);
    setWorkDir(TBDS::genDBGet(nodePath()+"Workdir").c_str());
    setIcoDir(TBDS::genDBGet(nodePath()+"IcoDir",icoDir()));
    setModDir(TBDS::genDBGet(nodePath()+"ModDir",modDir()));
    setSaveAtExit(atoi(TBDS::genDBGet(nodePath()+"SaveAtExit","0").c_str()));
    setSavePeriod(atoi(TBDS::genDBGet(nodePath()+"SavePeriod","0").c_str()));
}

void TSYS::load_()
{
    static bool first_load = true;

    bool cmd_help = cfgFileLoad();
    mess_info(nodePath().c_str(),_("Load!"));
    cfgPrmLoad();
    Mess->load(); //Завантажуємо повідомлення

    if( first_load )
    {
      //> Створюємо підсистему
      add( new TBDS() );
      add( new TSecurity() );
      add( new TTransportS() );
      add( new TProtocolS() );
      add( new TDAQS() );
      add( new TArchiveS() );
      add( new TSpecialS() );
      add( new TUIS() );
      add( new TModSchedul() );

      //> Завантажуємо модулі
      modSchedul().at().load();
      if( !modSchedul().at().loadLibS() )
```

```
      {
          mess_err(nodePath().c_str(),_("Жоден модуль не завантажений. Ваша
конфігурація перервана!"));
          stop();
      }

      //> Завантажуємо базу даних першої  підсистеми моніторингу продуктивності
мережі на базі SNMP/NMS
      db().at().load();
      if( !cmd_help ) modSchedul().at().modifG();     // Для перевантаження
спроби від бази даних

      //> Друге завантаження для завантаження від родової БД
      Mess->load();
      cfgPrmLoad();
    }

    //> Пряме завантаження підсистем та модулів
    vector<string> lst;
    list(lst);
    for( unsigned i_a=0; i_a < lst.size(); i_a++ )
      try { at(lst[i_a]).at().load(); }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(nodePath().c_str(),_("Помилка завантаження підсистеми
моніторингу продуктивності мережі на базі SNMP/NMS '%s'."),lst[i_a].c_str());
      }

    if( cmd_help ) stop();
    first_load = false;
}

void TSYS::save_( )
{
    char buf[STR_BUF_LEN];

    mess_info(nodePath().c_str(),_("Save!"));

    //> Системні параметри
    getcwd(buf,sizeof(buf));
    TBDS::genDBSet(nodePath()+"StName",mName,"root",TBDS::UseTranslate);
    TBDS::genDBSet(nodePath()+"Workdir",buf);
    TBDS::genDBSet(nodePath()+"IcoDir",icoDir());
    TBDS::genDBSet(nodePath()+"ModDir",modDir());
    TBDS::genDBSet(nodePath()+"SaveAtExit",TSYS::int2str(saveAtExit()));
    TBDS::genDBSet(nodePath()+"SavePeriod",TSYS::int2str(savePeriod()));

    Mess->save(); //Завантажуємо повідомлення
}

int TSYS::start( )
{
    vector<string> lst;
    list(lst);

    mess_info(nodePath().c_str(),_("Start!"));
    for(unsigned i_a=0; i_a < lst.size(); i_a++)
      try { at(lst[i_a]).at().subStart(); }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(nodePath().c_str(),_("Помилка запуску підсистеми моніторингу
продуктивності мережі на базі SNMP/NMS'%s'."),lst[i_a].c_str());
      }

    cfgFileScan( true );

    mess_info(nodePath().c_str(),_("Завершення запуску!"));
```

```
    unsigned int i_cnt = 1;
    mStopSignal = 0;
    while(!mStopSignal)
    {
      //> CPU підрахунок частоти
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))     clkCalc( );

      //> Кофігураційний файл змін періодичних перевірок
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))     cfgFileScan( );

      //> Періодична загальна перевірка бібліотек
      if(modSchedul( ).at().chkPer() && !(i_cnt%(modSchedul(
).at().chkPer()*1000/STD_WAIT_DELAY)))
          modSchedul( ).at().libLoad(modDir(),true);

      //> Періодичний запис змін до  БД
      if(savePeriod() && !(i_cnt%(savePeriod()*1000/STD_WAIT_DELAY))) save();

      //> Кофігураційний файл зберігає необхідні зміни
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))     cfgFileSave();

      //> Викликаємо підсистему кожні 10 сек.
      if(!(i_cnt%(10*1000/STD_WAIT_DELAY)))
          for(unsigned i_a=0; i_a < lst.size(); i_a++)
            try { at(lst[i_a]).at().perSYSCall(i_cnt/(1000/STD_WAIT_DELAY)); }
            catch(TError err) { mess_err(err.cat.c_str(),"%s",err.mess.c_str());
}

      usleep(STD_WAIT_DELAY*1000);
      i_cnt++;
    }

    mess_info(nodePath().c_str(),_("Stop!"));
    if(saveAtExit() || savePeriod())     save();
    cfgFileSave();
    for(int i_a=lst.size()-1; i_a >= 0; i_a--)
      try { at(lst[i_a]).at().subStop(); }
      catch(TError err)
      {
          mess_err(err.cat.c_str(),"%s",err.mess.c_str());
          mess_err(nodePath().c_str(),_("Помилка остановки підсистеми
моніторингу продуктивності мережі на базі SNMP/NMS'%s'."),lst[i_a].c_str());
      }

    return mStopSignal;
}

void TSYS::stop( )
{
    mStopSignal = SIGUSR1;
}

bool TSYS::chkSelDB( const string& wDB,  bool isStrong )
{
    if(selDB().empty() && !isStrong) return true;
    if(SYS->selDB() == TBDS::realDBName(wDB)) return true;
    return false;
}

void TSYS::sighandler( int signal )
{
    switch(signal)
    {
      case SIGINT:
          SYS->mStopSignal=signal;
          break;
      case SIGTERM:
```

```
          mess_warning(SYS->nodePath().c_str(),_("Отриманий сигнал переривання
роботи. Сервер зупиняється!"));
          SYS->mStopSignal=signal;
          break;
      case SIGFPE:
          mess_warning(SYS->nodePath().c_str(),_("Виключення плаваючої крапки
спіймане!"));
            exit(1);
          break;
      case SIGCHLD:
      {
          int status;
          pid_t pid = wait(&status);
          if(!WIFEXITED(status) && pid > 0)
            mess_info(SYS->nodePath().c_str(),_("Вивільнено процес-
потомок%d!"),pid);
          break;
      }
      case SIGPIPE:
          //mess_warning(SYS->nodePath().c_str(),_("Сигнал переривання
PIPE!"));
          break;
      case SIGSEGV:
          mess_emerg(SYS->nodePath().c_str(),_("Сигнал помилки від сегменту!"));
          break;
      case SIGABRT:
          mess_emerg(SYS-
>nodePath().c_str(),_("WebSNMP_NMS_Application_Centric_Infrastructure перервала
роботу!"));
          break;
      case SIGALRM:     break;
      default:
          mess_warning(SYS->nodePath().c_str(),_("Невизначений
сигнал%d!"),signal);
    }
}

void TSYS::cfgFileScan( bool first )
{
    struct stat f_stat;

    if(stat(cfgFile().c_str(),&f_stat) != 0) return;
    bool up = false;
    if(rootCfgFl != cfgFile() || rootFlTm != f_stat.st_mtime) up = true;
    rootCfgFl = cfgFile();
    rootFlTm = f_stat.st_mtime;

    if(up && !first)
    {
      modifG();
      setSelDB("<cfg>");
      load();
      setSelDB("");
    }
}

int64_t TSYS::curTime( )
{
    timeval cur_tm;
    gettimeofday(&cur_tm,NULL);
    return (int64_t)cur_tm.tv_sec*1000000 + cur_tm.tv_usec;
}

bool TSYS::eventWait( bool &m_mess_r_stat, bool exempl, const string &loc,
time_t tm )
{
    time_t t_tm, s_tm;

    t_tm = s_tm = time(NULL);
```

```
    while( m_mess_r_stat != exempl )
    {
      time_t c_tm = time(NULL);
      //Контролюємо перерву
      if( tm && ( c_tm > s_tm+tm) )
      {
          mess_crit(loc.c_str(),_("Timeouted !!!"));
          return true;
      }
      //Створюємо повідомлення
      if( c_tm > t_tm+1 )  //1sec
      {
          t_tm = c_tm;
          mess_info(loc.c_str(),_("Чекаємо подію..."));
      }
      usleep(STD_WAIT_DELAY*1000);
    }
    return false;
}

string TSYS::strSepParse( const string &path, int level, char sep, int *off )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0;
    size_t t_dir;

    if(an_dir >= (int)path.size()) return "";
    while(true)
    {
      t_dir = path.find(sep,an_dir);
      if( t_dir == string::npos )
      {
          if( off ) *off = path.size();
          return (t_lev == level) ? path.substr(an_dir) : "";
      }
      else if( t_lev == level )
      {
          if( off ) *off = t_dir+1;
          return path.substr(an_dir,t_dir-an_dir);
      }
      an_dir = t_dir+1;
      t_lev++;
    }
    return "";
}

string TSYS::strParse( const string &path, int level, const string &sep, int
*off, bool mergeSepSymb )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0;
    size_t t_dir;

    if(an_dir >= (int)path.size() || sep.empty()) return "";
    while(true)
    {
      t_dir = path.find(sep,an_dir);
      if( t_dir == string::npos )
      {
          if( off ) *off = path.size();
          return (t_lev == level) ? path.substr(an_dir) : "";
      }
      else if( t_lev == level )
      {
          if( off ) *off = t_dir+sep.size();
          return path.substr(an_dir,t_dir-an_dir);
      }
      if( mergeSepSymb && sep.size() == 1 )
```

```
          for(an_dir = t_dir; an_dir < (int)path.size() && path[an_dir] ==
sep[0]; ) an_dir++;
      else an_dir = t_dir+sep.size();
      t_lev++;
    }
    return "";
}

string TSYS::strLine( const string &str, int level, int *off )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0, edLnSmbSz = 1;
    size_t t_dir;

    if(an_dir >= (int)str.size()) return "";
    while(true)
    {
      for(t_dir = an_dir; t_dir < str.size(); t_dir++)
          if(str[t_dir] == '\x0D' || str[t_dir] == '\x0A')
          { edLnSmbSz = (str[t_dir] == '\x0D' && ((t_dir+1) < str.size()) &&
str[t_dir+1] == '\x0A') ? 2 : 1; break; }
      if(t_dir >= str.size())
      {
          if(off) *off = str.size();
          return (t_lev==level) ? str.substr(an_dir) : "";
      }
      else if(t_lev == level)
      {
          if(off) *off = t_dir+edLnSmbSz;
          return str.substr(an_dir,t_dir-an_dir);
      }
      an_dir = t_dir+edLnSmbSz;
      t_lev++;
    }
    return "";
}

string TSYS::pathLev( const string &path, int level, bool encode, int *off )
{
    int an_dir = off ? *off : 0;
    int t_lev = 0;
    size_t t_dir;

    //> Перший роздільний прохід
    while(an_dir < (int)path.size() && path[an_dir]=='/') an_dir++;
    if(an_dir >= (int)path.size()) return "";
    //> Шлях рівня процесу
    while(true)
    {
      t_dir = path.find("/",an_dir);
      if( t_dir == string::npos )
      {
          if( off ) *off = path.size();
          return (t_lev == level) ? ( encode ?
TSYS::strDecode(path.substr(an_dir),TSYS::PathEl) : path.substr(an_dir) ) : "";
      }
      else if( t_lev == level )
      {
          if( off ) *off = t_dir;
          return encode ? TSYS::strDecode(path.substr(an_dir,t_dir-
an_dir),TSYS::PathEl) : path.substr(an_dir,t_dir-an_dir);
      }
      an_dir = t_dir;
      t_lev++;
      while(an_dir < (int)path.size() && path[an_dir]=='/') an_dir++;
    }
}

string TSYS::path2sepstr( const string &path, char sep )
```

```
{
    string rez, curv;
    int off = 0;
    while( !(curv=TSYS::pathLev(path,0,false,&off)).empty() )
      rez+=curv+sep;
    if(!rez.empty())    rez.resize(rez.size()-1);

    return rez;
}

string TSYS::sepstr2path( const string &str, char sep )
{
    string rez, curv;
    int off = 0;
    while( !(curv=TSYS::strSepParse(str,0,sep,&off)).empty() )
      rez+="/"+curv;

    return rez;
}

string TSYS::strEncode( const string &in, TSYS::Code tp, const string &symb )
{
    int i_sz;
    string sout;

    switch(tp)
    {
      case TSYS::PathEl:
          sout = in;
          for( i_sz = 0; i_sz < (int)sout.size(); i_sz++ )
            switch( sout[i_sz] )
            {
                case '/': sout.replace(i_sz,1,"%2f"); i_sz+=2; break;
                case '%': sout.replace(i_sz,1,"%25"); i_sz+=2; break;
            }
          break;
      case TSYS::HttpURL:
          sout = in;
          for( i_sz = 0; i_sz < (int)sout.size(); i_sz++ )
            switch( sout[i_sz] )
            {
                case '%': sout.replace(i_sz,1,"%25"); i_sz+=2; break;
                case ' ': sout.replace(i_sz,1,"%20"); i_sz+=2; break;
                case '\t': sout.replace(i_sz,1,"%09"); i_sz+=2; break;
                default:
                  if( sout[i_sz]&0x80 )
                  {
                      char buf[4];
                      snprintf(buf,sizeof(buf),"%%%02X",(unsigned
char)sout[i_sz]);
                      sout.replace(i_sz,1,buf);
                      i_sz+=2;
                      break;
                  }
            }
          break;
      case TSYS::Html:
          sout.reserve(in.size()+10);
          for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
            switch( in[i_sz] )
            {
                case '>':     sout+="&gt;";     break;
                case '<':     sout+="&lt;";     break;
                case '"':     sout+="&quot;";   break;
                case '&':     sout+="&amp;";    break;
                case '\'':    sout+="&apos;";   break;
                default:      sout+=in[i_sz];
            }
          break;
```

```
        case TSYS::JavaSc:
            sout.reserve(in.size()+10);
            for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
              switch( in[i_sz] )
              {
                  case '\n':    sout+="\\n";      break;
                  default:      sout+=in[i_sz];
              }
            break;
        case TSYS::SQL:
            sout.reserve(in.size()+10);
            for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
              switch( in[i_sz] )
              {
                  case '\'':    sout+="\\'";      break;
                  case '\"':    sout+="\\\"";     break;
                  case '`':     sout+="\\`";      break;
                  case '\\':    sout+="\\\\";     break;
                  default:      sout+=in[i_sz];
              }
            break;
        case TSYS::Custom:
            sout.reserve(in.size()+10);
            for( i_sz = 0; i_sz < (int)in.size(); i_sz++ )
            {
              unsigned i_smb;
              for(i_smb = 0; i_smb < symb.size(); i_smb++)
                  if(in[i_sz] == symb[i_smb])
                  {
                    char buf[4];
                    sprintf(buf,"%%%02X",(unsigned char)in[i_sz]);
                    sout += buf;
                    break;
                  }
              if(i_smb >= symb.size()) sout += in[i_sz];
            }
            break;
        case TSYS::base64:
        {
            sout.reserve(in.size()+in.size()/4+in.size()/57+10);
            const char *base64alph =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
            for( i_sz = 0; i_sz < (int)in.size(); i_sz+=3 )
            {
              if(i_sz && !(i_sz%57))  sout.push_back('\n');
              sout.push_back(base64alph[(unsigned char)in[i_sz]>>2]);
              if((i_sz+1) >= (int)in.size())
              {
                  sout.push_back(base64alph[((unsigned char)in[i_sz]&0x03)<<4]);
                  sout += "==";
              }
              else
              {
                  sout.push_back(base64alph[(((unsigned
char)in[i_sz]&0x03)<<4)|((unsigned char)in[i_sz+1]>>4)]);
                  if((i_sz+2) >= (int)in.size())
                  {
                    sout.push_back(base64alph[((unsigned
char)in[i_sz+1]&0x0F)<<2]);
                    sout.push_back('=');
                  }
                  else
                  {
                    sout.push_back(base64alph[(((unsigned
char)in[i_sz+1]&0x0F)<<2)|((unsigned char)in[i_sz+2]>>6)]);
                    sout.push_back(base64alph[(unsigned char)in[i_sz+2]&0x3F]);
                  }
              }
            }
```

```
            break;
        }
        case TSYS::FormatPrint:
            sout = in;
            for(i_sz = 0; i_sz < (int)sout.size(); i_sz++)
              if(sout[i_sz] == '%') { sout.replace(i_sz,1,"%%"); i_sz++; }
            break;
        case TSYS::oscdID:
            sout.reserve(in.size());
            for(i_sz = 0; i_sz < (int)in.size(); i_sz++)
              switch(in[i_sz])
              {
                  case ' ': case '/': case '\\': case '&': case '(':
                  case ')': case '[': case ']': case '!': case '~':
                  case '`': case '@': case '%': case '^': case '-':
                  case '+': case '=': case '*': case '{': case '}':
                  case ':': case ';': case '"': case '\'': case '<':
                  case '>': case '?': case '.': case ',':
                    sout+="_";  break;
                  default:      sout+=in[i_sz];
              }
            break;
        case TSYS::Bin:
        {
            string svl, evl;
            sout.reserve(in.size());
            for(int off = 0; (svl=TSYS::strSepParse(in,0,'\n',&off)).size(); )
              for(int offE = 0; (evl=TSYS::strSepParse(svl,0,' ',&offE)).size(); )
                  sout+=(char)strtol(evl.c_str(),NULL,16);
            break;
        }
        case TSYS::Reverse:
            for(i_sz = in.size()-1; i_sz >= 0; i_sz--) sout += in[i_sz];
            break;
        case TSYS::ShieldSimb:
            sout.reserve(in.size());
            for(i_sz = 0; i_sz < (int)in.size(); i_sz++)
              if(in[i_sz] == '\\' && i_sz < ((int)in.size()-1))
              {
                  switch(in[i_sz+1])
                  {
                    case 'a':   sout += '\a';     break;
                    case 'b':   sout += '\b';     break;
                    case 'f':   sout += '\f';     break;
                    case 'n':   sout += '\n';     break;
                    case 'r':   sout += '\r';     break;
                    case 't':   sout += '\t';     break;
                    case 'v':   sout += '\v';     break;
                    case 'x': case 'X':
                        if((i_sz+3) < (int)in.size() && isxdigit(in[i_sz+2]) &&
isxdigit(in[i_sz+3]))
                        { sout +=
(char)strtol(in.substr(i_sz+2,2).c_str(),NULL,16); i_sz += 2; }
                        else sout += in[i_sz+1];
                              break;
                    default:
                              if((i_sz+3) < (int)in.size() && in[i_sz+1] >= '0' &&
in[i_sz+1] <= '7' &&
                                                              in[i_sz+2] >= '0' &&
in[i_sz+2] <= '7' &&
                                                              in[i_sz+3] >= '0' &&
in[i_sz+3] <= '7')
                              { sout +=
(char)strtol(in.substr(i_sz+1,3).c_str(),NULL,8); i_sz += 2; }
                              else sout += in[i_sz+1];
                  }
                  i_sz++;
              }else sout += in[i_sz];
            break;
```

```
    }
    return sout;
}

unsigned char TSYS::getBase64Code(unsigned char asymb)
{
    switch(asymb)
    {
      case 'A' ... 'Z': return asymb-(unsigned char)'A';
      case 'a' ... 'z': return 26+asymb-(unsigned char)'a';
      case '0' ... '9': return 52+asymb-(unsigned char)'0';
      case '+':         return 62;
      case '/':         return 63;
    }
    return 0;
}

string TSYS::strDecode( const string &in, TSYS::Code tp )
{
    unsigned i_sz;
    string sout;

    switch(tp)
    {
      case TSYS::PathEl: case TSYS::HttpURL: case TSYS::Custom:
          sout.reserve(in.size());
          for(i_sz = 0; i_sz < in.size(); i_sz++)
            switch(in[i_sz])
            {
                case '%':
                  if(i_sz+2 < in.size())
                  {
                      sout += (char)strtol(in.substr(i_sz+1,2).c_str(),NULL,16);
                      i_sz += 2;
                  }else sout += in[i_sz];
                  break;
                default: sout += in[i_sz];
            }
          break;
      case TSYS::base64:
          sout.reserve(in.size());
          for( i_sz = 0; i_sz < in.size(); )
          {
            if(in[i_sz] == '\n')    i_sz+=sizeof('\n');
            if((i_sz+3) < in.size())
              if( in[i_sz+1] != '=' )
              {
                char w_code1 = TSYS::getBase64Code(in[i_sz+1]);

      sout.push_back((TSYS::getBase64Code(in[i_sz])<<2)|(w_code1>>4));
                if( in[i_sz+2] != '=' )
                {
                    char w_code2 = TSYS::getBase64Code(in[i_sz+2]);
                    sout.push_back((w_code1<<4)|(w_code2>>2));
                    if( in[i_sz+3] != '=' )

      sout.push_back((w_code2<<6)|TSYS::getBase64Code(in[i_sz+3]));
                }
              }
            i_sz+=4;
          }
          break;
      case TSYS::Bin:
          sout.reserve(in.size());
          for( i_sz = 0; i_sz < in.size(); i_sz++ )
            sout += TSYS::strMess(((i_sz+1)%16)?"%0.2x ":"%0.2x\n",(unsigned
char)in[i_sz]);
          break;
      default: sout = in;     break;
```

```
    }

    return sout;
}

string TSYS::strCompr( const string &in, int lev )
{
    z_stream strm;

    if( in.empty() )    return "";

    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;

    if( deflateInit(&strm,lev) != Z_OK ) return "";

    uLongf comprLen = deflateBound(&strm,in.size());
    char out[comprLen];

    strm.next_in = (Bytef*)in.data();
    strm.avail_in = (uInt)in.size();
    strm.next_out = (Bytef*)out;
    strm.avail_out = comprLen;

    if( deflate(&strm, Z_FINISH) != Z_STREAM_END )
    {
      deflateEnd(&strm);
      return "";
    }

    comprLen = strm.total_out;

    deflateEnd(&strm);

    return string(out,comprLen);
}

string TSYS::strUncompr( const string &in )
{
    int ret;
    z_stream strm;
    unsigned char out[STR_BUF_LEN];
    string rez;

    if( in.empty() )    return "";

    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;

    if( inflateInit(&strm) != Z_OK )      return "";

    strm.avail_in = in.size();
    strm.next_in = (Bytef*)in.data();
    do
    {
      strm.avail_out = sizeof(out);
      strm.next_out = out;
      ret=inflate(&strm,Z_NO_FLUSH);
      if( ret == Z_STREAM_ERROR || ret == Z_NEED_DICT || ret == Z_DATA_ERROR ||
ret == Z_MEM_ERROR )
          break;
      rez.append((char*)out,sizeof(out)-strm.avail_out);
    } while( strm.avail_out == 0 );

    inflateEnd(&strm);

    if( ret != Z_STREAM_END ) return "";
```

```
    return rez;
}

float TSYS::floatLE(float in)
{
#if __BYTE_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      float f;
      struct
        {
          unsigned int mantissa:23;
          unsigned int exponent:8;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_be.f = in;
    ieee754_le.ieee.mantissa  = ieee754_be.ieee.mantissa;
    ieee754_le.ieee.exponent  = ieee754_be.ieee.exponent;
    ieee754_le.ieee.negative  = ieee754_be.ieee.negative;

    return ieee754_le.f;
#endif

    return in;
}

float TSYS::floatLErev(float in)
{
#if __BYTE_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      float f;
      struct
        {
          unsigned int mantissa:23;
          unsigned int exponent:8;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_le.f = in;
    ieee754_be.ieee.mantissa  = ieee754_le.ieee.mantissa;
    ieee754_be.ieee.exponent  = ieee754_le.ieee.exponent;
    ieee754_be.ieee.negative  = ieee754_le.ieee.negative;

    return ieee754_be.f;
#endif

    return in;
}

double TSYS::doubleLE(double in)
{
#if __BYTE_ORDER == __BIG_ENDIAN || __FLOAT_WORD_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      double d;
      struct
        {
          unsigned int mantissa1:32;
          unsigned int mantissa0:20;
          unsigned int exponent:11;
          unsigned int negative:1;
```

```
      } ieee;
    } ieee754_le;

    ieee754_be.d = in;
    ieee754_le.ieee.mantissa0 = ieee754_be.ieee.mantissa0;
    ieee754_le.ieee.mantissa1 = ieee754_be.ieee.mantissa1;
    ieee754_le.ieee.exponent  = ieee754_be.ieee.exponent;
    ieee754_le.ieee.negative  = ieee754_be.ieee.negative;

    return ieee754_le.d;
#endif

    return in;
}

double TSYS::doubleLErev(double in)
{
#if __BYTE_ORDER == __BIG_ENDIAN || __FLOAT_WORD_ORDER == __BIG_ENDIAN
    ieee754_double ieee754_be;
    union ieee754_le
    {
      double d;
      struct
        {
          unsigned int mantissa1:32;
          unsigned int mantissa0:20;
          unsigned int exponent:11;
          unsigned int negative:1;
      } ieee;
    } ieee754_le;

    ieee754_le.d = in;
    ieee754_be.ieee.mantissa0 = ieee754_le.ieee.mantissa0;
    ieee754_be.ieee.mantissa1 = ieee754_le.ieee.mantissa1;
    ieee754_be.ieee.exponent  = ieee754_le.ieee.exponent;
    ieee754_be.ieee.negative  = ieee754_le.ieee.negative;

    return ieee754_be.d;
#endif

    return in;
}

long TSYS::HZ()
{
    return sysconf(_SC_CLK_TCK);
}

bool TSYS::cntrEmpty( )
{
    ResAlloc res( nodeRes(), false );
    return mCntrs.empty();
}

double TSYS::cntrGet( const string &id )
{
    ResAlloc res( nodeRes(), false );
    map<string,double>::iterator icnt = mCntrs.find(id);
    if( icnt == mCntrs.end() )      return 0;
    return icnt->second;
}

void TSYS::cntrSet( const string &id, double vl )
{
    ResAlloc res( nodeRes(), true );
    mCntrs[id] = vl;
}
```

```
void TSYS::taskCreate( const string &path, int priority, void
*(*start_routine)(void *), void *arg, int wtm, pthread_attr_t *pAttr, bool
*startSt )
{
    int detachStat = 0;
    pthread_t procPthr;
    pthread_attr_t locPAttr, *pthr_attr;
    map<string,STask>::iterator ti;

    ResAlloc res(taskRes, true);
    for(time_t c_tm = time(NULL); mTasks.find(path) != mTasks.end(); )
    {
      if(time(NULL) >= (c_tm+wtm)) throw TError(nodePath().c_str(),_("Завдання
'%s' вже присутнє!"),path.c_str());
      res.release();
      usleep(10000);
      res.request(true);
    }
    STask &htsk = mTasks[path];
    htsk.path = path;
    htsk.task = start_routine;
    htsk.taskArg = arg;
    htsk.flgs = 0;
    res.release();

    if(pAttr) pthr_attr = pAttr;
    else
    {
      pthr_attr = &locPAttr;
      pthread_attr_init(pthr_attr);
    }
    pthread_attr_setinheritsched(pthr_attr, PTHREAD_EXPLICIT_SCHED);
    struct sched_param prior;
    prior.sched_priority = 0;

    int policy = SCHED_OTHER;
#if __GLIBC_PREREQ(2,4)
    if(priority < 0)    policy = SCHED_BATCH;
#endif
    if(priority > 0 /*&& SYS->user() == "root"*/)    policy = SCHED_RR;
    pthread_attr_setschedpolicy(pthr_attr, policy);
    prior.sched_priority =
vmax(sched_get_priority_min(policy),vmin(sched_get_priority_max(policy),priority
));
    pthread_attr_setschedparam(pthr_attr,&prior);

    try
    {
      pthread_attr_getdetachstate(pthr_attr,&detachStat);
      if(detachStat == PTHREAD_CREATE_DETACHED) htsk.flgs |= STask::Detached;
      int rez = pthread_create(&procPthr, pthr_attr, taskWrap, &htsk);
      if(rez == EPERM)
      {
        mess_warning(nodePath().c_str(),_("No permission for create real-time
policy. Default thread is created!"));
        policy = SCHED_OTHER;
        pthread_attr_setschedpolicy(pthr_attr, policy);
        prior.sched_priority = 0;
        pthread_attr_setschedparam(pthr_attr,&prior);
        rez = pthread_create(&procPthr, pthr_attr, taskWrap, &htsk);
      }
      if(!pAttr) pthread_attr_destroy(pthr_attr);

      if(rez) throw TError(nodePath().c_str(), _("Завдання створило
помилку%d."), rez);

      //> Чекаємо закінчення ініціалізації структури потоку для не відривних
завдань
      while(!(htsk.flgs&STask::Detached) && !htsk.thr) pthread_yield();
```

```
      //> Чекаємо запуск статусу
      for(time_t c_tm = time(NULL); !(htsk.flgs&STask::Detached) && startSt &&
!(*startSt); )
      {
          if(time(NULL) >= (c_tm+wtm)) throw
TError(nodePath().c_str(),_("Завдання '%s' запуск відкладений!"),path.c_str());
          usleep(STD_WAIT_DELAY *1000);
      }
    }
    catch(TError)
    {
      res.request(true);
      mTasks.erase(path);
      res.release();
      throw;
    }
}

void TSYS::taskDestroy( const string &path, bool *endrunCntr, int wtm, bool
noSignal )
{
    ResAlloc res(taskRes, false);
    map<string,STask>::iterator it = mTasks.find(path);
    if(it == mTasks.end()) return;
    pthread_t thr = it->second.thr;
    res.release();

    if(endrunCntr) *endrunCntr = true;
    if(!noSignal) pthread_kill(thr, SIGALRM);

    //> Чекаємо завершення завдання та повторюємо відправлення SIGALRM
    time_t t_tm, s_tm;
    t_tm = s_tm = time(NULL);
    while(!(it->second.flgs&STask::FinishTask))
    {
        if(!noSignal) pthread_kill(thr, SIGALRM);
      time_t c_tm = time(NULL);
        //Контролюємо перерву
        if(wtm && (c_tm > (s_tm+wtm)))
        {
            mess_crit((nodePath()+path+": stop").c_str(),_("Timeouted !!!"));
          throw TError(nodePath().c_str(),_(«завдання '%s' is not
stopped!"),path.c_str());
        }
      //Створюємо повідомлення
        if(c_tm > t_tm+1)  //1sec
        {
            t_tm = c_tm;
            mess_info((nodePath()+path+": stop").c_str(),_("Чекаємо подію..."));

        }
        usleep(STD_WAIT_DELAY*1000);
    }

    if(!(it->second.flgs&STask::Detached)) pthread_join(thr, NULL);

    res.request(true);
    mTasks.erase(it);
}

void *TSYS::taskWrap( void *stas )
{
    //> Беремо тимчасову структуру завдання
    STask *tsk = (STask *)stas;
    pthread_setspecific(TSYS::sTaskKey, tsk);

    //> Запам'ятовуємо параметри виклику
    void *(*wTask) (void *) = tsk->task;
    void *wTaskArg = tsk->taskArg;
```

```
    //> Отримуємо поточні політику і пріорітет
    int policy;
    struct sched_param param;
    pthread_getschedparam(pthread_self(), &policy, &param);
    tsk->policy = policy;
    tsk->prior = param.sched_priority;

#if __GLIBC_PREREQ(2,4)
    //> Отримуємо і завантажуємо CPU установлення
    if(SYS->multCPU() && !(tsk->flgs & STask::Detached))
    {
      tsk->cpuSet = TBDS::genDBGet(SYS->nodePath()+"CpuSet:"+tsk->path);
      cpu_set_t cpuset;
      CPU_ZERO(&cpuset);
      string sval;
      bool cpuSetOK = false;
      for(int off = 0; (sval=TSYS::strParse(tsk->cpuSet,0,":",&off)).size();
cpuSetOK = true)
          CPU_SET(atoi(sval.c_str()),&cpuset);
      if(cpuSetOK) pthread_setaffinity_np(pthread_self(), sizeof(cpu_set_t),
&cpuset);
    }
    else if(SYS->multCPU() && (tsk->flgs & STask::Detached)) tsk->cpuSet = "NA";
#endif

    //> Закінчуємо установки та ініціалізуємо індікатор закінчення
    tsk->tid = syscall(SYS_gettid);
    tsk->thr = pthread_self();

    //> Викликаємо робочі завдання
    void *rez = NULL;
    try { rez = wTask(wTaskArg); }
    catch(TError err)
    {
      mess_err(err.cat.c_str(),err.mess.c_str());
      mess_err(SYS->nodePath().c_str(),_("Завдання %u несподівано закінчено
виключенням."),tsk->thr);
    }

    //> Відмічаємо закінчення завдання
    tsk->flgs |= STask::FinishTask;

    //> Переміщуємо об'єкти завдання окремо
    if(tsk->flgs & STask::Detached) SYS->taskDestroy(tsk->path, NULL);

    return rez;
}

void TSYS::taskSleep( int64_t per, time_t cron )
{
    struct timespec sp_tm;
    STask *stsk = (STask*)pthread_getspecific(sTaskKey);

    if(!cron)
    {
      if(!per) per = 1000000000;
      clock_gettime(CLOCK_REALTIME,&sp_tm);
      int64_t end_tm = (int64_t)sp_tm.tv_sec*1000000000+sp_tm.tv_nsec;
      int64_t pnt_tm = (end_tm/per + 1)*per;
      do
      {
          sp_tm.tv_sec = pnt_tm/1000000000; sp_tm.tv_nsec = pnt_tm%1000000000;
          if(clock_nanosleep(CLOCK_REALTIME,TIMER_ABSTIME,&sp_tm,NULL))
      return;
          clock_gettime(CLOCK_REALTIME,&sp_tm);
      }while(((int64_t)sp_tm.tv_sec*1000000000+sp_tm.tv_nsec) < pnt_tm);

      if(stsk)
```

```
      {
          stsk->tm_beg = stsk->tm_per;
          stsk->tm_end = end_tm;
          stsk->tm_per = (int64_t)sp_tm.tv_sec*1000000000+sp_tm.tv_nsec;
      }
    }
    else
    {
      time_t end_tm = time(NULL);
      while(time(NULL) < cron && usleep(1000000) == 0) ;
      if(stsk)
      {
          stsk->tm_beg = stsk->tm_per;
          stsk->tm_end = 1000000000ll*end_tm;
          stsk->tm_per = 1000000000ll*time(NULL);
      }
    }
}

time_t TSYS::cron( const string &vl, time_t base )
{
    string cronEl, tEl;
    int vbeg, vend, vstep, vm;

    time_t ctm = base?base:time(NULL);
    struct tm ttm;
    localtime_r(&ctm,&ttm);
    ttm.tm_sec = 0;

reload:
    bool isReload = false;

    //> Хвилини check
    cronEl = TSYS::strSepParse(vl,0,' ');
    vm = 200;
    for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
    {
      vbeg = vend = -1; vstep = 0;
      sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
      if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=0; vend=59; }
      if(vend < 0) vm = vmin(vm,vbeg+((ttm.tm_min>=vbeg)?60:0));
      else if((vbeg=vmax(0,vbeg)) < (vend=vmin(59,vend)))
      {
          if(ttm.tm_min < vbeg) vm = vmin(vm,vbeg);
          else if((vstep>1 && ttm.tm_min >= (vbeg+((vend-vbeg)/vstep)*vstep)) ||
(vstep <= 0 && ttm.tm_min >= vend))
            vm = vmin(vm,vbeg+60);
          else if(vstep>1 ) vm = vmin(vm, vbeg + vstep*(((ttm.tm_min+1)-
vbeg)/vstep + ((((ttm.tm_min+1)-vbeg)%vstep)?1:0)));
          else vm = vmin(vm, ttm.tm_min+1);
      }
      if(vm == ttm.tm_min+1) break;
    }
    ttm.tm_min = vm;
    mktime(&ttm);

    //> Перевіряємо час
    cronEl = TSYS::strSepParse(vl,1,' ');
    vm = 200;
    for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
    {
      vbeg = vend = -1; vstep = 0;
      sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
      if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=0; vend=23; }
      if(vend < 0) vm = vmin(vm,vbeg+((ttm.tm_hour>vbeg)?24:0));
      else if((vbeg=vmax(0,vbeg)) < (vend=vmin(23,vend)))
      {
          if(ttm.tm_hour < vbeg) vm = vmin(vm,vbeg);
```

```
          else if((vstep>1 && ttm.tm_hour > (vbeg+((vend-vbeg)/vstep)*vstep)) ||
(vstep <= 0 && ttm.tm_hour > vend))
            vm = vmin(vm,vbeg+24);
          else if(vstep>1 ) vm = vmin(vm, vbeg + vstep*((ttm.tm_hour-vbeg)/vstep
+ (((ttm.tm_hour-vbeg)%vstep)?1:0)));
          else vm = vmin(vm, ttm.tm_hour);
      }
      if(vm == ttm.tm_hour) break;
    }
    isReload = (vm != 200 && ttm.tm_hour!=vm);
    ttm.tm_hour = vm;
    mktime(&ttm);
    if(isReload)  { ttm.tm_min = -1; goto reload; }

    //> Перевіряємо день
    cronEl = TSYS::strSepParse(vl,2,' ');
    string cronElw = TSYS::strSepParse(vl,4,' ');
    vm = 200;
    if(cronEl != "*")
      for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
      {
          vbeg = vend = -1; vstep = 0;
          sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
          if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=1; vend=31; }
          if(vend < 0) vm = vmin(vm,vbeg+((ttm.tm_mday>vbeg)?31:0));
          else if((vbeg=vmax(1,vbeg)) < (vend=vmin(31,vend)))
          {
            if(ttm.tm_mday < vbeg) vm = vmin(vm,vbeg);
            else if((vstep>1 && ttm.tm_mday > (vbeg+((vend-vbeg)/vstep)*vstep))
|| (vstep <= 0 && ttm.tm_mday > vend))
                vm = vmin(vm,vbeg+31);
            else if(vstep>1 ) vm = vmin(vm, vbeg + vstep*((ttm.tm_mday-
vbeg)/vstep + (((ttm.tm_mday-vbeg)%vstep)?1:0)));
            else vm = vmin(vm, ttm.tm_mday);
          }
          if(vm == ttm.tm_mday) break;
       }
    if(cronEl == "*" || (cronElw != "*" && !cronElw.empty()))
      for(int eoff = 0; (tEl=TSYS::strSepParse(cronElw,0,',',&eoff)).size(); )
      {
          vbeg = vend = -1; vstep = 0;
          sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
          if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=0; vend=6; }
          if(vend < 0) vm = vmin(vm, ttm.tm_mday - ttm.tm_wday +
vbeg+((ttm.tm_wday>vbeg)?7:0));
          else if((vbeg=vmax(0,vbeg)) < (vend=vmin(6,vend)))
          {
            if(ttm.tm_wday < vbeg) vm = vmin(vm, ttm.tm_mday - ttm.tm_wday +
vbeg);
            else if((vstep>1 && ttm.tm_wday > (vbeg+((vend-vbeg)/vstep)*vstep))
|| (vstep <= 0 && ttm.tm_wday > vend))
                vm = vmin(vm, ttm.tm_mday - ttm.tm_wday + vbeg+7);
            else if(vstep>1) vm = vmin(vm, ttm.tm_mday - ttm.tm_wday + vbeg +
vstep*((ttm.tm_wday-vbeg)/vstep + (((ttm.tm_wday-vbeg)%vstep)?1:0)));
            else vm = vmin(vm, ttm.tm_mday);
          }
          if(vm == ttm.tm_mday) break;
      }
    isReload = (vm!=200 && ttm.tm_mday!=vm);
    if(vm <= 31) ttm.tm_mday = vm;
    else { ttm.tm_mday = vm-31; ttm.tm_mon++; }
    mktime(&ttm);
    if(isReload)  { ttm.tm_min = -1; ttm.tm_hour = 0; goto reload; }

    //> Перевіряємо місяць
    cronEl = TSYS::strSepParse(vl,3,' ');
    vm = 200;
    for(int eoff = 0; (tEl=TSYS::strSepParse(cronEl,0,',',&eoff)).size(); )
    {
```

```
      vbeg = vend = -1; vstep = 0;
      sscanf(tEl.c_str(),"%d-%d/%d",&vbeg,&vend,&vstep);
      if(vbeg < 0) { sscanf(tEl.c_str(),"*/%d",&vstep); vbeg=1; vend=12; }
      if(vend < 0) vm = vmin(vm,vbeg+(((ttm.tm_mon+1)>vbeg)?12:0));
      else if((vbeg=vmax(1,vbeg)) < (vend=vmin(12,vend)))
      {
          if((ttm.tm_mon+1) < vbeg) vm = vmin(vm,vbeg);
          else if((vstep>1 && (ttm.tm_mon+1) > (vbeg+((vend-vbeg)/vstep)*vstep))
|| (vstep <= 0 && (ttm.tm_mon+1) > vend))
            vm = vmin(vm,vbeg+12);
          else if(vstep>1) vm = vmin( vm, vbeg + vstep*(((ttm.tm_mon+1)-
vbeg)/vstep + ((((ttm.tm_mon+1)-vbeg)%vstep)?1:0)));
          else vm = vmin(vm, ttm.tm_mon+1);
      }
      if(vm == (ttm.tm_mon+1)) break;
    }
    isReload = (vm!=200 && ttm.tm_mon!=(vm-1));
    ttm.tm_mon = vm-1;
    mktime(&ttm);
    if(isReload)  { ttm.tm_min = -1; ttm.tm_hour = 0; ttm.tm_mday = 1; goto
reload; }

    return mktime(&ttm);
}

TVariant TSYS::objFuncCall( const string &iid, vector<TVariant> &prms, const
string &user )
{
    // int message(string cat, int level, string mess) - форматуємо системне
повідомлення  <mess> з категорією <cat>, рівнем <level>
    //  cat - повідомлення категорії
    //  level - повідомлення рівня
    //  mess - повідомлення тексту
    if(iid == "message" && prms.size() >= 3)    { message(
prms[0].getS().c_str(), (TMess::Type)prms[1].getI(), "%s",
prms[2].getS().c_str() ); return 0; }
    // int messDebug(string cat, string mess) - форматуємо системне повідомлення
<mess> з категорією <cat> і відповідний рівень
    //  cat - повідомлення категорії
    //  mess - повідомлення тексту
    if(iid == "messDebug" && prms.size() >= 2) { mess_debug(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messInfo" && prms.size() >= 2)  { mess_info(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messNote" && prms.size() >= 2)  { mess_note(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messWarning" && prms.size() >= 2){ mess_warning(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messErr" && prms.size() >= 2)    { mess_err(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messCrit" && prms.size() >= 2)  { mess_crit(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messAlert" && prms.size() >= 2) { mess_alert(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    if(iid == "messEmerg" && prms.size() >= 2) { mess_emerg(
prms[0].getS().c_str(), "%s", prms[1].getS().c_str() ); return 0; }
    // string system(string cmd, bool noPipe = false) - викликаємо консольні
команди <cmd> для повернення у ОС  результатів з каналів
    //  cmd - текс команди
    //  noPipe - результат блокують для другорядного виклику
    if(iid == "system" && prms.size() >= 1)
    {
      if(prms.size() >= 2 && prms[1].getB()) return
system(prms[0].getS().c_str());
      FILE *fp = popen(prms[0].getS().c_str(),"r");
      if(!fp) return string("");

      char buf[STR_BUF_LEN];
      string rez;
```

```
      for(int r_cnt = 0; (r_cnt=fread(buf,1,sizeof(buf),fp)); )
          rez.append(buf,r_cnt);

      pclose(fp);
      return rez;
    }
    // string fileRead( string file ) - Повертаємо <file> контент у рядку.
    if(iid == "fileRead" && prms.size() >= 1)
    {
      char buf[STR_BUF_LEN];
      string rez;
        int hd = open(prms[0].getS().c_str(),O_RDONLY);
      if(hd != -1)
      {
          for(int len = 0; (len=read(hd,buf,sizeof(buf))) > 0; )
rez.append(buf,len);
          close(hd);
      }
      return rez;
    }
    // int fileWrite( string file, string str, bool append = false ) - Записуємо
<str> до <file>, переміщуємо представлення, або <append>.
    //        Return wrote bytes count.
    if(iid == "fileWrite" && prms.size() >= 2)
    {
      int wcnt = 0, wflags = O_WRONLY|O_CREAT|O_TRUNC;
      string val = prms[1].getS();
      if(prms.size() >= 3 && prms[2].getB()) wflags = O_WRONLY|O_CREAT|O_APPEND;
        int hd = open(prms[0].getS().c_str(), wflags, 0664);
      if(hd != -1)
      {
          wcnt = write(hd,val.data(),val.size());
          close(hd);
      }
      return wcnt;
    }
    // XMLNodeObj XMLNode(string name = "") - створюємо XML об'єкт вузлів з
іменем <name>
    //  name - XML ім'я вузлу
    if(iid == "XMLNode") return new XMLNodeObj((prms.size()>=1) ? prms[0].getS()
: "");
    // string cntrReq(XMLNodeObj req, string stat = "") - запит інтерфейсу
управління до системи моніторингу продуктивності мережі на базі SNMP/NMS через
XML
    //  req - запити XML вузлів
    //  stat - переміщуємо WebSNMP_NMS_Application_Centric_Infrastructure-робочу
станцію для запиту
    if(iid == "cntrReq" && prms.size() >= 1)
    {
      XMLNode req;
      if(!dynamic_cast<XMLNodeObj*>(prms[0].getO())) return string(_("1:Запит не
об'єктний!"));
      ((XMLNodeObj*)prms[0].getO())->toXMLNode(req);
      string path = req.attr("path");
      if(prms.size() < 2 || prms[1].getS().empty())
      {
          req.setAttr("user",user);
          cntrCmd(&req);
      }
      else
      {
          req.setAttr("path","/"+prms[1].getS()+path);
          transport().at().cntrIfCmd(req,"cntrReq");
          req.setAttr("path",path);
      }
      ((XMLNodeObj*)prms[0].getO())->fromXMLNode(req);
      return string("0");
    }
```

```
    // string sleep(int tm, int ntm = 0) – викликаємо завдання переходу до
сплячого режиму через <tm> секунд та <ntm> наносекунд.
    //  tm - чекаємо цей час у секундах
    //  ntm - чекаємо цей час у наносекундах
    if(iid == "sleep" && prms.size() >= 1)
    {
      struct timespec sp_tm;
      sp_tm.tv_sec = prms[0].getI();
      sp_tm.tv_nsec = (prms.size() >= 2) ? prms[1].getI() : 0;
      int rez = clock_nanosleep(CLOCK_REALTIME,0,&sp_tm,NULL);
      return rez;
    }
    // int time(int usec) – повертаємо абсолютний час у секундах від 1/1/1970 та
в мікросекундах, якщо <usec> задано
    //  usec - microсекунди of time
    if(iid == "time")
    {
      if(prms.empty()) return (int)time(NULL);
      int64_t tm = curTime();
      prms[0].setI(tm%1000000); prms[0].setModify();
      return (int)(tm/1000000);
    }
    // int localtime(int fullsec, int sec, int min, int hour, int mday, int
month, int year, int wday, int yday, int isdst)
    //      - повертаємо повну дату, базуємо на абсолютному часі у секундах
<fullsec> від 1.1.1970
    //  fullsec – час джерела у секундах від 1.1.1970
    //  sec - секунди
    //  min - хвилини
    //  hour - часи
    //  mday - дні місяця
    //  month - місяці
    //  year - роки
    //  wday - дні неділі
    //  yday - дні року
    //  isdst - відмітка про літній час
    if(iid == "localtime" && prms.size() >= 2)
    {
      time_t tm_t = prms[0].getI();
      struct tm tm_tm;
      localtime_r(&tm_t,&tm_tm);

      prms[1].setI(tm_tm.tm_sec); prms[1].setModify();
      if(prms.size() >= 3)    { prms[2].setI(tm_tm.tm_min); prms[2].setModify();
}
      if(prms.size() >= 4)    { prms[3].setI(tm_tm.tm_hour);
prms[3].setModify(); }
      if(prms.size() >= 5)    { prms[4].setI(tm_tm.tm_mday);
prms[4].setModify(); }
      if(prms.size() >= 6)    { prms[5].setI(tm_tm.tm_mon); prms[5].setModify();
}
      if(prms.size() >= 7)    { prms[6].setI(1900+tm_tm.tm_year);
prms[6].setModify(); }
      if(prms.size() >= 8)    { prms[7].setI(tm_tm.tm_wday);
prms[7].setModify(); }
      if(prms.size() >= 9)    { prms[8].setI(tm_tm.tm_yday);
prms[8].setModify(); }
      if(prms.size() >= 10)   { prms[9].setI(tm_tm.tm_isdst);
prms[9].setModify(); }
      return 0;
    }
    // string strftime(int sec, string form = "%Y-%m-%d %H:%M:%S") – перетворює
абсолютний час  <sec> у рядок формату <form>
    //  sec – час у секундах від 1.1.1970
    //  form – вихідний форматований рядок
    if(iid == "strftime" && !prms.empty())
    {
      time_t tm_t = prms[0].getI();
      struct tm tm_tm;
```

```
      localtime_r(&tm_t,&tm_tm);
      char buf[1000];
      int rez = strftime(buf, sizeof(buf), (prms.size()>=2) ?
prms[1].getS().c_str() : "%Y-%m-%d %H:%M:%S", &tm_tm);
      return (rez>0) ? string(buf,rez) : "";
    }
    // int strptime(string str, string form = "%Y-%m-%d %H:%M:%S") - повертає
час  у секундах від of 1/1/1970,
    //      базується на запису рядку часу <str>, відповідно до вказаного
шаблону  <form>
    //  str - джерело часу у рядку
    //  form - рядки часу у форматі POSIX-функцій "strptime"
    if(iid == "strptime" && !prms.empty())
    {
      struct tm stm;
      stm.tm_isdst = -1;
      strptime(prms[0].getS().c_str(), (prms.size()>=2) ? prms[1].getS().c_str()
: "%Y-%m-%d %H:%M:%S", &stm);
      return (int)mktime(&stm);
    }
    // int cron(string cronreq, int base = 0) - повертає час , планується у
форматі стандартного Cron <cronreq>,
    //      початок від основного часу<base> або від поточного, якщо основа не
вказана
    //  cronreq - розповсюджений у стандартному форматі Cron
    //  base - основний час
    if(iid == "cron" && !prms.empty())
      return (int)cron(prms[0].getS(), (prms.size()>=2) ? prms[1].getI() : 0);
    // string strFromCharCode(int char1, int char2, int char3, ...) - створює
рядок з кодових символів
    //  char1, char2. char3 - кодові символи
    if(iid == "strFromCharCode")
    {
      string rez;
      for(unsigned i_p = 0; i_p < prms.size(); i_p++)
          rez += (unsigned char)prms[i_p].getI();
      return rez;
    }
    // string strCodeConv( string src, string fromCP, string toCP ) - Текстовий
рядок перекодовує з кодової сторінки <fromCP> до кодової сторінки <toCP>.
    //  src - source text;
    //  fromCP - з кодової сторінки , порожньої для внутрішньої кодової сторінки
;
    //  toCP - до кодової сторінки , порожньої для внутрішньої кодової сторінки
.
    if(iid == "strCodeConv" && prms.size() >= 3)
      return Mess->codeConv((prms[1].getS().size() ? prms[1].getS() : Mess-
>charset()),
                  (prms[2].getS().size() ? prms[2].getS() : Mess->charset()),
prms[0].getS());

    return TCntrNode::objFuncCall(iid,prms,user);
}

void TSYS::cntrCmdProc( XMLNode *opt )
{
    char buf[STR_BUF_LEN];

    //Беремо сторінку інформації
    if(opt->name() == "info")
    {
      TCntrNode::cntrCmdProc(opt);
      snprintf(buf,sizeof(buf),_("%s station:
\"%s\""),PACKAGE_NAME,name().c_str());
      ctrMkNode("SNMP_NMS_Application_Centric_Infrastructure_cntr",opt,-
1,"/",buf,R_R_R_);
      if(ctrMkNode("branches",opt,-1,"/br","",R_R_R_))
          ctrMkNode("grp",opt,-
1,"/br/sub_",_("Subsystem"),R_R_R_,"root","root",1,"idm","1");
```

```
      if(TUIS::icoPresent(id())) ctrMkNode("img",opt,-1,"/ico","",R_R_R_);
      if(ctrMkNode("area",opt,-1,"/gen",_("Station"),R_R_R_))
      {
          ctrMkNode("fld",opt,-
1,"/gen/id",_("ID"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-
1,"/gen/stat",_("Station"),RWRWR_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-
1,"/gen/prog",_("Program"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-
1,"/gen/ver",_("Version"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/host",_("Ім'я
хосту"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/user",_("Користувач системи моніторингу
продуктивності мережі на базі SNMP/NMS"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/sys",_("Операційна
система"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/frq",_("Частота
(MHZ)"),R_R_R_,"root","root",1,"tp","real");
          ctrMkNode("fld",opt,-1,"/gen/clk_res",_("Значення годинника реального
часу"),R_R_R_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/in_charset",_("Внутрішній набір
символів"),R_R___,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/config",_("Кофігураційний
файл"),R_R___,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/workdir",_("Робоча
директорія"),RWRW__,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/icodir",_("Директорія
іконок"),RWRW__,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/moddir",_("Директорія
модулів"),RWRW__,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/wrk_db",_("Робоча база
даних"),RWRWR_,"root","root",4,"tp","str","dest","select","select","/db/list",
            "help",_("Адрес робочої бази даних у форматі [<БД module>.<БД
name>].\n Змінюємо ці поля якщо необхідно зберегти або завантажити усю систему з
іншої  БД."));
          ctrMkNode("fld",opt,-1,"/gen/saveExit",_("Збереження змін у системі
перед виходом"),RWRWR_,"root","root",2,"tp","bool",
            "help",_("Обираємо автоматичне збереження системи моніторингу
продуктивності мережі на базі SNMP/NMS до  БД перед виходом."));
          ctrMkNode("fld",opt,-1,"/gen/savePeriod",_("Збереження періоду роботи
у системі "),RWRWR_,"root","root",2,"tp","dec",
            "help",_("Використовуємо нульовий період (секунди) для періодичного
збереження  змін частин системи моніторингу продуктивності мережі на базі
SNMP/NMS у БД."));
          ctrMkNode("fld",opt,-
1,"/gen/lang",_("Language"),RWRWR_,"root","root",1,"tp","str");
          ctrMkNode("fld",opt,-1,"/gen/baseLang",_("Базова мова тексту
змінних"),RWRWR_,"root","root",5,"tp","str","len","2","dest","sel_ed","select","
/gen/baseLangLs",
            "help",_("Мультимовність для змінного тексту підтримки для вибору
базової мови."));
          if(ctrMkNode("area",opt,-1,"/gen/mess",_("Повідомлення"),R_R_R_))
          {
            ctrMkNode("fld",opt,-1,"/gen/mess/lev",_("Найменший
рівень"),RWRWR_,"root","root",3,
                "tp","dec","len","1","help",_("Повідомлення найменшого рівня для
відображення процесів, які відбуваються у системі."));
            ctrMkNode("fld",opt,-1,"/gen/mess/log_sysl",_("To
syslog"),RWRWR_,"root","root",1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/gen/mess/log_stdo",_("To
stdout"),RWRWR_,"root","root",1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/gen/mess/log_stde",_("To
stderr"),RWRWR_,"root","root",1,"tp","bool");
            ctrMkNode("fld",opt,-1,"/gen/mess/log_arch",_("До
архіву"),RWRWR_,"root","root",1,"tp","bool");
          }
      }
      if(ctrMkNode("area",opt,-1,"/subs",_("Підсистема")))
```

```
          ctrMkNode("list",opt,-1,"/subs/br",_("Підсистеми моніторингу
продуктивності мережі на базі
SNMP/NMS"),R_R_R_,"root","root",3,"idm","1","tp","br","br_pref","sub_");
      if(ctrMkNode("area",opt,-1,"/tasks",_("Tasks"),R_R___))
          if(ctrMkNode("table",opt,-
1,"/tasks/tasks",_("Завдання"),RWRW__,"root","root",2,"key","path",
            "help",!multCPU()?"":_("Для CPU встановлюємо рядок номерів
процесорів використання, відокремлений символом':'.\n"
                                   "CPU починаємо з 0.")))
          {
            ctrMkNode("list",opt,-
1,"/tasks/tasks/path",_("Path"),R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/thrd",_("Потік"),R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/tid",_("TID"),R_R___,"root","root",1,"tp","dec");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/stat",_("Статус"),R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/plc",_("Політика"),R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/tasks/tasks/prior",_("Prior."),R_R___,"root","root",1,"tp","dec");
#if __GLIBC_PREREQ(2,4)
            if(multCPU())
                ctrMkNode("list",opt,-1,"/tasks/tasks/cpuSet",_("CPU
встановлення"),RWRW__,"root","root",1,"tp","str");
#endif
          }
      if( !cntrEmpty() && ctrMkNode("area",opt,-1,"/cntr",_("Країна")) )
          if( ctrMkNode("table",opt,-
1,"/cntr/cntr",_("Counters"),R_R___,"root","root") )
          {
            ctrMkNode("list",opt,-
1,"/cntr/cntr/id","ID",R_R___,"root","root",1,"tp","str");
            ctrMkNode("list",opt,-
1,"/cntr/cntr/vl",_("Value"),R_R___,"root","root",1,"tp","real");
          }
      if( ctrMkNode("area",opt,-1,"/hlp",_("Help"),R_R___) )
          ctrMkNode("fld",opt,-1,"/hlp/g_help",_("Опції
допомоги"),R_R___,"root","root",3,"tp","str","cols","90","rows","10");
      return;
    }

    //Процес управління на сторінці
    string a_path = opt->attr("path");
    if(a_path == "/ico" && ctrChkNode(opt))
    {
      string itp;
      opt->setText(TSYS::strEncode(TUIS::icoGet(id(),&itp),TSYS::base64));
      opt->setAttr("tp",itp);
    }
    else if(a_path == "/gen/host" && ctrChkNode(opt))opt->setText(host());
    else if(a_path == "/gen/sys" && ctrChkNode(opt))
    {
      utsname ubuf; uname(&ubuf);
      opt->setText(string(ubuf.sysname)+"-"+ubuf.release);
    }
    else if(a_path == "/gen/user" && ctrChkNode(opt))opt->setText(mUser);
    else if(a_path == "/gen/prog" && ctrChkNode(opt))opt->setText(PACKAGE_NAME);
    else if(a_path == "/gen/ver" && ctrChkNode(opt))  opt->setText(VERSION);
    else if(a_path == "/gen/id" && ctrChkNode(opt))   opt->setText(id());
    else if(a_path == "/gen/stat")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(name());
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setName(opt->text());
    }
    else if(a_path == "/gen/frq" && ctrChkNode(opt))  opt-
>setText(TSYS::real2str((float)sysClk()/1000000.,6));
    else if(a_path == "/gen/clk_res" && ctrChkNode(opt))
```

```
    {
      struct timespec tmval;
      clock_getres(CLOCK_REALTIME,&tmval);
      opt->setText(TSYS::time2str(1e-3*tmval.tv_nsec));//
TSYS::real2str((float)tmval.tv_nsec/1000000.,4));
    }
    else if(a_path == "/gen/in_charset" && ctrChkNode(opt))opt->setText(Mess-
>charset());
    else if(a_path == "/gen/config" && ctrChkNode(opt))          opt-
>setText(mConfFile);
    else if(a_path == "/gen/wrk_db" )
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt-
>setText(mWorkDB);
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setWorkDB(opt-
>text());
    }
    else if(a_path == "/gen/saveExit")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(
int2str(saveAtExit()) );
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setSaveAtExit(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/gen/savePeriod")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(
int2str(savePeriod()) );
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))setSavePeriod(
atoi(opt->text().c_str()) );
    }
    else if(a_path == "/gen/workdir")
    {
      if(ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD))opt-
>setText(workDir());
      if(ctrChkNode(opt,"set",R_R___,"root","root",SEC_WR))setWorkDir(opt-
>text().c_str());
    }
    else if(a_path == "/gen/icodir")
    {
      if(ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD))opt-
>setText(icoDir());
      if(ctrChkNode(opt,"set",R_R___,"root","root",SEC_WR))setIcoDir(opt-
>text().c_str());
    }
    else if(a_path == "/gen/moddir")
    {
      if(ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD))opt-
>setText(modDir());
      if(ctrChkNode(opt,"set",R_R___,"root","root",SEC_WR))setModDir(opt-
>text().c_str());
    }
    else if(a_path == "/gen/lang")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(Mess-
>lang());
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLang(opt-
>text());
    }
    else if(a_path == "/gen/baseLang")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText(Mess-
>lang2CodeBase());
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess-
>setLang2CodeBase(opt->text());
    }
    else if(a_path == "/gen/baseLangLs" && ctrChkNode(opt))
    {
      opt->childAdd("el")->setText(Mess->lang2Code());
```

```
      if(!Mess->lang2CodeBase().empty() && Mess->lang2CodeBase() != Mess-
>lang2Code())
          opt->childAdd("el")->setText(Mess->lang2CodeBase());
      opt->childAdd("el")->setText("");
    }
    else if(a_path == "/gen/mess/lev")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt-
>setText(TSYS::int2str(Mess->messLevel()));
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess-
>setMessLevel(atoi(opt->text().c_str()));
    }
    else if(a_path == "/gen/mess/log_sysl")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x01)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x01:Mess->logDirect()&(~0x01) );
    }
    else if(a_path == "/gen/mess/log_stdo")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x02)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x02:Mess->logDirect()&(~0x02) );
    }
    else if(a_path == "/gen/mess/log_stde")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x04)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x04:Mess->logDirect()&(~0x04) );
    }
    else if(a_path == "/gen/mess/log_arch")
    {
      if(ctrChkNode(opt,"get",RWRWR_,"root","root",SEC_RD))opt->setText((Mess-
>logDirect()&0x08)?"1":"0");
      if(ctrChkNode(opt,"set",RWRWR_,"root","root",SEC_WR))Mess->setLogDirect(
atoi(opt->text().c_str())?Mess->logDirect()|0x08:Mess->logDirect()&(~0x08) );
    }
    else if((a_path == "/br/sub_" || a_path == "/subs/br") &&
ctrChkNode(opt,"get",R_R_R_,"root","root",SEC_RD))
    {
      vector<string> lst;
      list(lst);
      for(unsigned i_a=0; i_a < lst.size(); i_a++)
          opt->childAdd("el")->setAttr("id",lst[i_a])-
>setText(at(lst[i_a]).at().subName());
    }
    else if(a_path == "/tasks/tasks")
    {
      if(ctrChkNode(opt,"get",RWRW__,"root","root"))
      {
          XMLNode *n_path     = ctrMkNode("list",opt,-
1,"/tasks/tasks/path","",R_R___,"root","root");
          XMLNode *n_thr      = ctrMkNode("list",opt,-
1,"/tasks/tasks/thrd","",R_R___,"root","root");
          XMLNode *n_tid      = ctrMkNode("list",opt,-
1,"/tasks/tasks/tid","",R_R___,"root","root");
          XMLNode *n_stat     = ctrMkNode("list",opt,-
1,"/tasks/tasks/stat","",R_R___,"root","root");
          XMLNode *n_plc      = ctrMkNode("list",opt,-
1,"/tasks/tasks/plc","",R_R___,"root","root");
          XMLNode *n_prior    = ctrMkNode("list",opt,-
1,"/tasks/tasks/prior","",R_R___,"root","root");
          XMLNode *n_cpuSet   = (multCPU() ? ctrMkNode("list",opt,-
1,"/tasks/tasks/cpuSet","",RWRW__,"root","root") : NULL);

          ResAlloc res(taskRes,false);
```

```
          for(map<string,STask>::iterator it = mTasks.begin(); it !=
mTasks.end(); it++)
          {
            if(n_path)  n_path->childAdd("el")->setText(it->first);
            if(n_thr)   n_thr->childAdd("el")->setText(TSYS::uint2str(it-
>second.thr));
            if(n_tid)   n_tid->childAdd("el")->setText(TSYS::int2str(it-
>second.tid));
            if(n_stat)
            {
                int64_t tm_beg = 0, tm_end = 0, tm_per = 0;
                for(int i_tr = 0; tm_beg == tm_per && i_tr < 2; i_tr++)
                { tm_beg = it->second.tm_beg; tm_end = it->second.tm_end; tm_per
= it->second.tm_per; }
                XMLNode *cn = n_stat->childAdd("el");
                if(it->second.flgs&STask::FinishTask) cn->setText(_("Закінчено.
"));
                if(tm_beg && tm_beg < tm_per)
                  cn->setText(cn->text()+TSYS::strMess(_("Останій: %s.
Завантажено: %3.1f% (%s з %s)"),
                    time2str((time_t)(1e-9*tm_per),"%d-%m-%Y
%H:%M:%S").c_str(), 100*(double)(tm_end-tm_beg)/(double)(tm_per-tm_beg),
                    time2str(1e-3*(tm_end-tm_beg)).c_str(), time2str(1e-
3*(tm_per-tm_beg)).c_str()));
            }
            if(n_plc)
            {
                string plcVl = _("Стандартний");
                if(it->second.policy == SCHED_RR) plcVl = _(" Кругова система
");
#if __GLIBC_PREREQ(2,4)
                if(it->second.policy == SCHED_BATCH) plcVl = _("Style
\"batch\"");
#endif
                n_plc->childAdd("el")->setText(plcVl);
            }
            if(n_prior) n_prior->childAdd("el")->setText(TSYS::int2str(it-
>second.prior));
            if(n_cpuSet)      n_cpuSet->childAdd("el")->setText(it-
>second.cpuSet);
          }
      }
#if __GLIBC_PREREQ(2,4)
      if(multCPU() && ctrChkNode(opt,"set",RWRW__,"root","root",SEC_WR) && opt-
>attr("col") == "cpuSet")
      {
          ResAlloc res(taskRes,true);
          map<string,STask>::iterator it = mTasks.find(opt->attr("key_path"));
          if(it == mTasks.end()) throw TError(nodePath().c_str(),_("Не
представлене завдання '%s'."));
          if(it->second.flgs & STask::Detached) return;

          it->second.cpuSet = opt->text();

          cpu_set_t cpuset;
          CPU_ZERO(&cpuset);
          string sval;
          for(int off = 0; (sval=TSYS::strParse(it-
>second.cpuSet,0,":",&off)).size(); )
            CPU_SET(atoi(sval.c_str()),&cpuset);
          int rez = pthread_setaffinity_np(it->second.thr, sizeof(cpu_set_t),
&cpuset);
          res.release();
          TBDS::genDBSet(nodePath()+"CpuSet:"+it->first,opt->text());
          if(rez == EINVAL && opt->text().size()) throw
TError(nodePath().c_str(),_("Не встановлено жодних дозволених процесорів."));
          if(rez && opt->text().size()) throw TError(nodePath().c_str(),_("CPU
установки для потоку помилкові."));
      }
```

```
#endif
    }
    if(!cntrEmpty() && a_path == "/cntr/cntr" &&
ctrChkNode(opt,"get",R_R___,"root","root"))
    {
      XMLNode *n_id     = ctrMkNode("list",opt,-
1,"/cntr/cntr/id","",R_R___,"root","root");
      XMLNode *n_vl     = ctrMkNode("list",opt,-
1,"/cntr/cntr/vl","",R_R___,"root","root");

      ResAlloc res( nodeRes(), false );
      for(map<string,double>::iterator icnt = mCntrs.begin(); icnt !=
mCntrs.end(); icnt++)
      {
          if(n_id)      n_id->childAdd("el")->setText(icnt->first);
          if(n_vl)      n_vl->childAdd("el")->setText(TSYS::real2str(icnt-
>second));
      }
    }
    else if(a_path == "/hlp/g_help" &&
ctrChkNode(opt,"get",R_R___,"root","root",SEC_RD)) opt->setText(optDescr());
    else TCntrNode::cntrCmdProc(opt);
}
```