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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи контейнерної віртуалізації мережної інфраструктури для хостингу”

Виконав здобувач вищої освіти
IV курсу, групи КМ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Харитоненко О.О.
«____» _____ 2024 р.

Керівник проекту
канд. фіз.-мат. наук, доцент
_____ Якименко Н.М.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

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

Харитоненку Олександру Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи контейнерної віртуалізації мережної інфраструктури для хостингу*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 133-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи контейнерної віртуалізації мережної інфраструктури для хостингу*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Харитоненко О.О.
(прізвище та ініціали)

АНОТАЦІЯ

Харитоненко О.О. Програмне забезпечення системи контейнерної віртуалізації мережної інфраструктури для хостингу. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи контейнерної віртуалізації мережної інфраструктури для хостингу.

Метою розробки є програмне забезпечення системи контейнерної віртуалізації мережної інфраструктури для хостингу.

Результат роботи – програмна реалізація системи контейнерної віртуалізації мережної інфраструктури для хостингу.

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 з ОС Windows 10/11.

Програму розроблено в середовищі RAD Studio Delphi 10.4.

Ключові слова: комп'ютерна інженерія, віртуалізація мережної інфраструктури

ABSTRACT

Kharytonenko O.O. Network infrastructure container virtualization system software for hosting. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the system of container virtualization of the network infrastructure for hosting.

The purpose of the development is the software of the container virtualization system of the network infrastructure for hosting.

The result of the work is the software implementation of the system of container virtualization of the network infrastructure for hosting.

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

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

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the RAD Studio Delphi 10.4 environment.

Keywords: computer engineering, network infrastructure virtualization

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

DMA – Direct Memory Access;

DOS – дискова операційна система;

RS-232C – Recommended Standart 232 Version C, ревізія – EIA-232D;

USRT (Universal Synchronous Receiver/Transmitter) – універсальний синхронний прийомопередавач;

АЛП– арифметико–логічний пристрій;

АЦП – аналогово–цифровий перетворювач;

ВДТ – відеодісплейні термінали;

ЕЛТ – електронно–люмінісцентна трубка;

ЕОМ – електронна обчислювальна машина;

ЗПР– запит переривання;

ОС – операційна система;

ПДП – теж саме що і DMA;

ПЗП – постійний запам’ятовуючий пристрій;

ПЗП – постійний запом’ятовуючий пристрій;

ПК – персональний комп’ютер;

ППЗП – перепрограмувальний постійний запом’ятовуючий пристрій;

ПЧ – перетворювач частоти;

РА – регістр адреси;

РК – регістр команд;

РМП – регістр маски переривань;

РП – регістр пріоритетів;

ТД – технічна документація;

ТЗ – технічна задача;

ЦАП – цифро–аналоговий перетворювач;

ВСТУП

Актуальність теми. Контейнерна віртуалізація ще недавно сприймалася як якась дивина, у найкращому разі її розглядали як недорогий варіант створення інфраструктури для хостингу. Але сьогодні, коли завдяки хмарній революції на перший план вийшли такі вимоги до центрів обробки даних, як еластичність, масштабованість і висока обчислювальна щільність, контейнери стали предметом підвищеного інтересу – насамперед тому, що вони як не можна краще підходять для рішення названих завдань. Чому ж про їх раніше забували й чому згадали тепер?

Загалом віртуалізація – це мистецтво запуску однієї операційної системи поверх іншої. Історія її розвитку досить довга й багата. Задовго до гіпервізорів і UNIX-систем віртуалізація використовувалася в мейнфреймах для поділу різних операційних систем. Широке поширення в системах UNIX і Linux вона одержала лише на початку століття. В 2001 році компанія VMware випустила серверний продукт для віртуалізації на основі гіпервізора, який привернув увагу корпоративних замовників. Практично в той же самий час компанія Parallels представила рішення для контейнерної віртуалізації Virtuozzo, що заслужило визнання в провайдерів послуг хостингу. Такий поділ зберігався майже 12 років: гіпервізорній віртуалізації не вдавався завоювати скільки-небудь значимий шматок ринку хостингу, а контейнери не могли проникнути в корпоративний сегмент. Перелом намітився в 2013 році, коли новий розроблювач Docker привернув увагу представників бізнесу до переваг контейнерної технології.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи контейнерної віртуалізації мережної інфраструктури для хостингу.

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

– Огляд існуючих систем контейнерної віртуалізації мережної інфраструктури для хостингу.

– Дослідження системи контейнерної віртуалізації мережної інфраструктури для хостингу.

– Програмна реалізація системи контейнерної віртуалізації мережної інфраструктури для хостингу.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі контейнерної віртуалізації мережної інфраструктури для хостингу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи контейнерної віртуалізації мережної інфраструктури для хостингу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

В 2005 році компанія Google зайнялася завданням масового надання Web-сервісів, а саме шукала спосіб еластичного масштабування ресурсів у своєму центрі обробки даних, щоб кожний користувач мав можливість одержати достатній рівень сервісу в будь-який момент, незалежно від поточного завантаження, а ресурси, що залишилися, можна було використовувати для службових фонових завдань.

Попрацювавши із традиційної віртуалізацією, співробітники Google порахували її не підходящою для рішення цього завдання. Головною проблемою стали занадто великі втрати продуктивності (відповідно, щільність виявилася занадто низкою) і недостатньо еластичний відгук для динамічного переконфігурування системи під навантаження, що змінилося, для масового надання Web-сервісів.

Останній пункт дуже важливий, тому що заздалегідь пророчити, скільки запитів – десятки, сотні тисяч або навіть мільйони – будуть обслуговувати Web-сервіси, неможливо. Але користувачі завжди чекають негайного відгуку (а це означає, що різниця між натисканням кнопки й появою результату на екрані повинна бути непомітна для очей) незалежно від того, скільки саме інших людей у цей же момент працюють із сервісом. Середній час завантаження гіпервізорної віртуальної машини – десятки секунд, тому такий тип віртуалізації не підходить для цього завдання.

У той же самий час одна група розроблювачів експериментувала з Linux і концепцією, заснованою на механізмі cgroups – так звані контейнери процесів. Google найняла цих фахівців для роботи над контейнеризацією своїх ЦОД з метою рішення проблеми еластичності при масштабуванні. У січні 2008 року

частина технології cgroup, використовуваної Google, була перенесена в ядро Linux. Так народився проект Linu Containers (LXC). Тим часом Parallels випустила версію своєї віртуалізації Virtuozzo з відкритим вихідним кодом за назвою OpenVZ. В 2011 році Google і Parallels прийшли до угоди про співробітництво в області контейнерних технологій. Результатом став реліз ядра Linux версії 3.8, представлений в 2013 році. У ньому були об'єднані всі актуальні на той момент контейнерні технології для Linux, що дозволило уникнути повторення хворобливого поділу ядер, як у випадку з KVM і Xen.

1.2 Область застосування

Для хостинг-провайдерів основна перевага контейнерної віртуалізації полягає в щільності, що корпоративним клієнтам була не дуже-те потрібна. Для останніх віртуалізація пропонувалося як рішення проблеми низької завантаженості серверного встаткування (щоб не простоювало) і спосіб використання вільних обчислювальних ресурсів. А раз необхідності збільшувати навантаження не було, то й контейнери практично ігнорувалися корпоративним сегментом, на який доводиться 85% усього ринку віртуалізації.

Ситуація початку мінятися після 2010 року – з ростом обсягів хмарних обчислень. При цьому компанії зштовхнулися з такими ж труднощами, що й Google, – як домогтися еластичного масштабування ресурсів у центрах обробки даних до того ж забезпечити гарний рівень сервісу. І в цьому випадку в гіпервізорів час завантаження виявилося занадто повільним для забезпечення швидкого відгуку системи, що приводило до нездатності оперативно виділяти необхідні обсяги ресурсів. У той же час контейнери дозволяють обслуговувати більше (як уже говорилося – до трьох разів) клієнтських запитів без необхідності додавати встаткування. У результаті корпоративні замовники нарешті звернули увагу на контейнери.

Але виникли наступні проблеми: по-перше, співробітники багатьох компаній просто не знали, що можливо й інші варіанти віртуалізації, крім гіпервізорів, і не вміли працювати з іншою технологією, а по-друге, значні бюджети вже були витрачені на покупку й керування гіпервізорними рішеннями. У подібних умовах повний перехід на нову технологію – не краща ідея.

Така ситуація зберігалася до 2013 року, коли компанія-розроблювач Docker продемонструвала, як легко «упакувати» контейнеризований додаток в Linux і розгорнути його з можливістю масштабування прямо в dotCloud (сервіс Platform as a Service, PaaS від Docker). Підприємства зацікавилися даним підходом. Одночасно OpenStack пообіцяла об'єднати системи керування хмарами (Cloud Management) у єдину платформу, що охоплює обидві технології віртуалізації. Коли бізнес нарешті-те побачив можливість управляти своїми центрами обробки даних на основі гіпервізорів за допомогою єдиного інструмента, що паралельно дозволяє здійснювати масштабне розгортання контейнеризованих додатків, почався справжній бум.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи контейнерної віртуалізації мережної інфраструктури для хостингу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Контейнерна віртуалізація або віртуалізація на рівні операційної системи – це метод віртуалізації, при якому ядро операційної системи підтримує кілька ізольованих екземплярів простору користувача, замість одного. Це знижує накладні витрати й дозволяє використовувати віртуалізацію найбільше ефективно.

Контейнерна віртуалізація не використовує віртуальні машини, а створює віртуальне оточення із власним простором процесів і мережним стеком. Екземпляри просторів користувача (часто називані контейнерами або зонами) з погляду користувача повністю ідентичні реальному серверу, але вони у своїй роботі використовують один екземпляр ядра операційної системи. Для linux-систем, ця технологія може розглядатися як поліпшена реалізація механізму chroot. Ядро забезпечує повну ізольованість контейнерів, тому програми з різних контейнерів не можуть впливати один на одного.

Віртуалізація на рівні операційної системи дає значно кращу продуктивність, масштабованість, щільність розміщення, динамічне керування ресурсами, а також легкість в адмініструванні, чим в альтернативних рішень.

Найпоширеніші зараз OpenVZ, LXC, FreeBSD jail і Solaris Containers.

OpenVZ

OpenVZ – реалізація технології віртуалізації на рівні ОС на ядрі Linux. OpenVZ дозволяє на одному фізичному сервері запускати безліч ізольованих копій операційної системи, називаних «віртуальні частки сервери» (Virtual Private Servers, VPS) або «віртуальні середовища» (Virtual Environments, VE). Оскільки

OpenVZ базується на ядрі Linux, на відміну від віртуальних машин (напр. VMware, Parallels Desktop) або паравіртуалізаційних технологій (напр. Xen), у ролі «гостьових» систем можуть виступати тільки дистрибутиви Linux. Накладні витрати на віртуалізацію дуже малі, і падіння продуктивності становить усього 1–3 %, у порівнянні зі звичайними Linux-системами. OpenVZ поширюється на умовах ліцензії GNU GPL і складається з модифікованого ядра Linux і користувальницьких утиліт. OpenVZ є базовою платформою для Virtuozzo – пропрієтарного продукту Parallels.

LXC

LXC (Linux Containers) – система віртуалізації на рівні операційної системи для запуску декількох ізольованих екземплярів операційної системи Linux на одному вузлі. LXC заснована на технології cgroups, що входить у ядро Linux, починаючи з версії 2.6.29. Серед прикладів використання – застосування в PaaS-хостингу Heroku для ізоляції динамічних контейнерів (dynos).

Docker

Docker – це програмне забезпечення для автоматизації розгортання й керування додатками в середовищі віртуалізації LXC. Docker дозволяє «упакувати» додаток з усім його оточенням і залежностями в контейнер, що може бути перенесений на будь-який Linux-системі м\з підтримкою cgroups у ядрі, а також надає середовище по керуванню контейнерами. Docker, наприклад, використовується в хмарній платформі Cocaine.

Libcontainer

Недавно Docker, Parallels, Canonical, Google і RedHat домовилися про злиття центральних частин своїх контейнерних проектів в один – libcontainer – і спільному, погодженому його розвитку.

Libcontainer – це системна бібліотека, що надає гнучкий і досить абстрактний інтерфейс до ядерних контейнерних компонентів. Небагато технічних подробиць. У ядрі не існує такого поняття як "контейнер". Говорячи про контейнери, розроблювачі ядра мають на увазі кілька різних підсистем ядра,

які дозволяють, при правильному використанні, ізолювати додатки у віртуальних середовищах. Це, головним чином, cgroups і namespaces. Пряме використання ядерних інтерфейсів можливо, але досить нетривіально.

Бібліотека libcontainer покликана полегшити процедуру їхнього використання, надавши програмістам інтерфейс, у якому є більше "звичні" і "піднесені" поняття, такі як "контейнер", "обчислювальні ресурси", "віртуальна мережа" і т.п.

Крім розвитку контейнерної технології, Parallels бачить в Libcontainer ще й спосіб одержати цю технологію в проекті OpenStack. OpenStack – це набір програм, що дозволяє розвертати хмарну інфраструктуру й управляти нею. Дотепер OpenStack працював тільки з віртуальними машинами, але останнім часом, з розвитком контейнерної технології, співтовариство розроблювачів OpenStack все частіше й частіше згадує про те, що непогано було б створювати контейнери й у хмарах, керованих OpenStack-Ом.

Уже було почато кілька спроб досягти цього – RackSpace розробив розширення на основі контейнерів OpenVZ, але це рішення не було прийнято співтовариством. Docker надав розширення, засноване на своїй технології, але й воно не закріпилося в проекті. Як подальший розвиток розроблювачі Parallels, Docker, Canonical і, можливо, RackSpace спробують привнести контейнери в OpenStack через Libcontainer.

Чому все це зараз так важливо? Справа в тому, що контейнерна технологія стає ключовим компонентом у більшості дистрибутивів Linux. Більше того, ми всі частіше зауважуємо, що вона стає затребуваної в корпоративному середовищі, хоча традиційно вважалася долею хостинг-спільноти.

Оскільки ринок поступово переходить від простих хостингових послуг до хмарних, ми вважаємо важливим, щоб у замовників були всі необхідні для цього інструменти, у тому числі – OpenStack і Docker. Останній, зокрема, придасться для впакування ваших додатків у контейнер. Причини, по яких Parallels підтримує відразу обидва проекти, однакові: одержати технології Open Source, на

основі яких можна побудувати диференційовану пропозицію, що відповідає конкретним потребам замовників. За допомогою OpenStack, компонентів відкритої технології для хмарних інструментів і інструментів автоматизації, за допомогою Linux Foundation (і Linux у цілому) у компанії вже працює таке рішення, як Parallels Cloud Server.

Сьогодні розроблювачі Parallels допомагають створити нове покоління docker-програм, таких як контейнеризировані додатки, і за рахунок цього розширити кількість сценаріїв використання для контейнерів в обчислювальній індустрії. А libcontainer як технологія open source для споживання контейнерної віртуалізації на рівні деталізації – це погоджена (між Docker, LXC, Google і Parallels) можливість зробити це.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

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

виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

- Додана підтримка Metal Driver GPU для macOS і iOS.
- Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.
- Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.
- Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.
- Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.
- Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи контейнерної віртуалізації мережної інфраструктури для хостингу.

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

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

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

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

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

д) розробити рекомендації по організаційних та методичних заходах, які

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розроблювачі й системні адміністратори можуть, нарешті, погодитися в одному – контейнери дають розроблювачам можливість експериментувати без усяких ризиків і обмежень. Віртуальні контейнери – це каталоги (якщо ви прихильник Windows, можете називати їхніми папками), які безпечним образом ізольовані від іншого простору операційної системи. По суті, контейнер – це безпечна система для розробки, що разом з хостовою операційною системою використовує найбільш важливі файли, при цьому дозволяючи розроблювачеві створювати додаток, не побоюючись можливих негативних наслідків для хоста. Контейнер оснащений своєю власною IP-адресою, ідентифікатором, файловою системою (тільки кілька файлів «позичаються» у хоста) і рівнем виконання.

Всі вже звикли до того, що системні адміністратори й розроблювачі ПЗ постійно сперечаються один з одним із приводу прав користувачів, прав додатків, доступу із привілеями адміністратора, розміщення додатків і вимог до виділюваного дискового простору. Контейнерна віртуалізація знімає всі ці протиріччя, крім питання про дисковий простір. Системні адміністратори зберігають за собою привілей виділення місця на диску для контейнерів, але в інших питаннях розроблювачі адмініструють свій контейнер як хочуть. Вони можуть перезавантажувати контейнер, коли їм заманеться, установлювати кожне ПЗ й виконувати будь-які тести й експерименти, не побоюючись потурбувати хост.

Що ще робить контейнери настільки привабливими для розроблювачів – це можливість налагоджувати додаток рівно в тому же середовищі, у якому воно буде експлуатуватися. Традиційно налагодження ведеться в середовищі, що має ряд відмінностей від промислової – інша процесорна архітектура (наприклад,

менше ядер), відмінності у версіях базового ПЗ, інший графік установки відновлень.

Ці неприємні відмінності створюють проблеми в підтримці ПЗ, які здатні звести адміністратора з розуму. Розроблювачі воліють будувати додатка з використанням самих свіжих версій використовуваного ПЗ. Але адміністратори робітничих середовищ не дозволять ставити такі версії, тому що до них є питання в галузі стабільності й наявності уразливостей, і ці питання цілком справедливі, особливо до ПЗ на стадії бета-версії.

Однак застосування контейнерів знімає цілий ряд подібних складностей налагодження ПЗ на промисловій системі. Розроблювачі можуть використовувати самі свіжі версії потрібних інструментів і перевіряти стабільність їхньої роботи. Всі можливі уразливості виявляються «замкнені» у контейнері й не несуть ризиків для промислової системи.

Однією з найбільш популярних контейнерних систем є Docker. Це відкрита платформа, що дає розроблювачам і системним адміністраторам можливість створювати, поширювати й виконувати розподілені додатки. У контейнерах Docker додатка можуть працювати на будь-якій операційній системі. Використання Docker дозволяє прискорити розробку додатків і скоротити час їхньої підготовки до початку продажів користувачам.

Контейнери являють собою полегшену альтернативу традиційним віртуальним машинам. Розроблювачам вони дають можливість налагодження в промисловому середовищі, а системним адміністраторам – гарантію стабільності робітничого середовища. Контейнери вже досить давно існують на ринку, але зараз вони переживають зліт популярності завдяки низьким накладним витратам, високому рівню безпеки й стабільності, а також забезпеченню цілісності «операційної системи, що притулила їх».

Якщо ви розроблювач, то вам належить випробувати контейнери у своїй роботі. Якщо ви системний адміністратор, то вам варто дозволити використання цих ізольованих середовищ у підтримувані вами системах.

3.2 Розробка структурної схеми

Гіпервізор працює в такий спосіб: операційна система хоста емулює апаратне забезпечення, поверх якого вже запускаються гостьові операційні системи. Це означає, що взаємозв'язок між гостьовою й хостовой операційними системами треба «залізній» парадигмі: усе, що «уміє» робити встаткування, повинне бути доступно гостьовий ОС із боку хостовой. Навпроти, контейнери – це віртуалізація на рівні операційної системи, а не встаткування, тобто кожна гостьова ОС використовує те ж саме ядро (а в деяких випадках – і інші частини ОС), що й хостова. Це дає контейнерам велика перевага: вони менше й компактніше гіпервізорних гостьових середовищ, оскільки в них з хостом набагато більше загального.

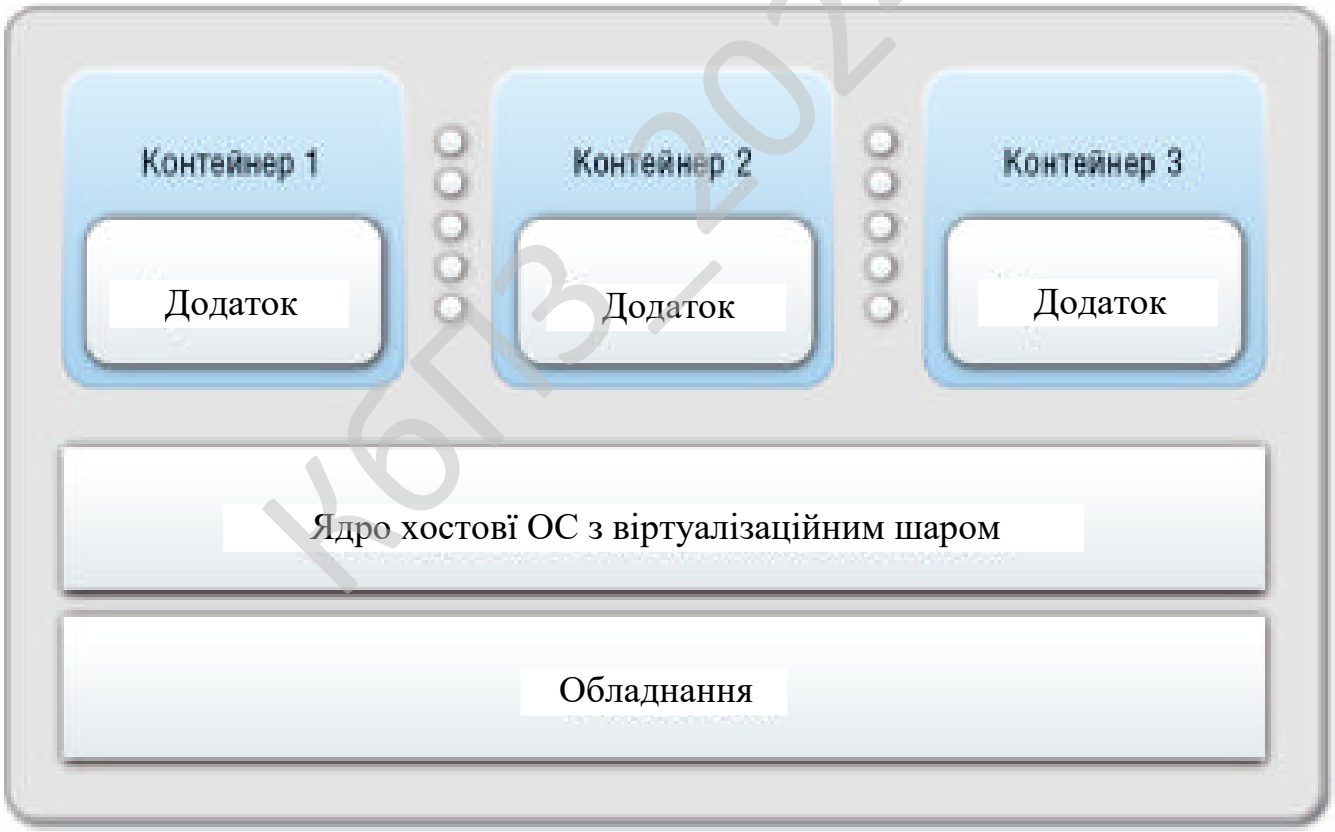

Рисунок 3.1 – Структурна схема системи

Інший великий плюс – значно більша ефективність контейнерної віртуалізації відносно спільного використання ресурсів, тому що для неї контейнери – це всього лише керовані ресурси. Приміром, коли Контейнер 1 і Контейнер 2 працюють із тим самим файлом, ядро хоста відкриває цей файл і розміщає сторінки з нього в сторінковий кеш ядра, які потім передаються Контейнеру 1 і Контейнеру 2: якщо обоє «хочуть» прочитати ті самі дані, вони одержують ту саму сторінку. Якщо ж гіпервізорним віртуальним машинам VM1 і VM2 треба виконати таку ж операцію, то спочатку сам хост відкриває запитуваний файл (створюючи сторінки у своєму сторінковому кеші), а потім ще й кожне з ядер VM1 і VM2 виконує аналогічну дію. Таким чином, у процесі читання машинами VM1 і VM2 того самого файли в пам'яті існує цілих три однакових сторінки (по однієї в сторінковому кеші хоста й у ядрах VM1 і VM2), тому що вони не «уміють» одночасно використовувати ту саму сторінку, як це роблять контейнери.

Справа навіть не стільки в «читанні» файлів, скільки в можливості використовувати одну копію файлів, що виконуються, і поділюваних бібліотек (shared libs) у різних контейнерах. У звичайній системі, якщо два або більше процеси звертаються до однієї й тій же поділюваній бібліотеці (наприклад, libc), її код присутній у пам'яті тільки в одному екземплярі. Це ставиться й до файлів, що виконуються, і до сегментів даних, що не модифікуються, що дозволяє істотно знизити вимоги до розміру оперативної пам'яті. Тому що контейнери використовують єдине ядро, вищеописаний механізм при деяких умовах поширюється й на них, що приводить, зокрема, до підвищення щільності їхнього розміщення, що і без цього механізму споконвічно вище, ніж у віртуальних машин, оскільки відсутні множинні копії ядра.

У результаті в контейнерів щільність (кількість віртуальних середовищ, які можна запустити на сервері) може бути до трьох разів вище, ніж у віртуальних машин, а на одному сервері цілком може розміщатися кілька сотень контейнерів. Настільки висока щільність – одна з головних причин популярності

контейнерів на ринку хостингу віртуальних виділених серверів (VPS). Якщо на тому самому сервері можна створити в три рази більше VPS, то розраховуючи на один VPS витрати знижуються на 66%, що для такого низкомаржинального бізнесу, як хостинг, іноді дорівнює різниці між збитковістю й прибутковістю.

Звичайно, не все ідеально у світі контейнерів. Наприклад, через спільне використання ядра на одному сервері не можна запускати різні гостьові ОС (наприклад, на системі з Linux-контейнерами неможливо запустити FreeBSD або Windows, але різні дистрибутиви Linux – скільки завгодно). Тому Windows і Linux на тому самому сервері (що для гіпервізорів не проблема) не працюють. Однак принаймні у випадку Linux цей недолік можна зм'якшити за рахунок використання інтерфейсів ABI і бібліотек: завдяки цьому з'являється можливість запускати на одному сервері контейнери з різними дистрибутивами Linux, але одночасно скорочується загальна для цих контейнерів частина ресурсів. У максимальному ж ступені переваги контейнерів проявляються в однорідному середовищі.

Як приклад нестандартного підходу до рішення завдань віртуалізації за допомогою контейнерів розглянемо віртуалізацію мережних функцій (Network Function Virtualization), досить затребувану технологію на ринку телекомунікацій. Традиційно при NFV з використанням гіпервізора мережна функція виконується в гостьовому ядрі й пропускає через себе весь мережний трафік, для чого застосовуються засоби віртуального драйвера. Останній, у свою чергу, витягає цей трафік безпосередньо з фізичного каналу передачі інформації (що відповідає першому рівню мережної моделі OSI). Завдяки такому підходу гостьова система може обробляти мережний трафік з ефективністю фізичного пристрою, перебуваючи при цьому у віртуальному середовищі під програмним контролем.

Згадану мережну функцію можна реалізувати в гостьовій системі через один або кілька стандартних мережних інтерфейсів. А замість впровадження драйвера можна спробувати приєднати контейнер до такий проксіруючої функції за допомогою засобів мережного інтерфейсу. Застосовувана в контейнерах

технологія за назвою «мережний простір імен» (network namespace) дозволяє перетворити мережний інтерфейс зі спільно використовуваного у виділюваний одному контейнеру. Тому, якщо їсти можливість впровадити спеціальний драйвер у загальне ядро операційної системи, мережний інтерфейс, з яким зв'язаний цей драйвер, можна як і раніше (як при використанні гіпервізора) передати в окремий контейнер за допомогою простору імен.

За допомогою такого спеціального драйвера контейнер одержує здатність обробляти мережний трафік у віртуальному середовищі, при цьому досягаються більша щільність віртуальних об'єктів і висока ефективність, адже контейнерна інфраструктура займає менший обсяг (у порівнянні з віртуальними машинами). Цю концепцію можна розвити: оскільки додаток у контейнері може користуватися сервісами фізичної операційної системи, кожне із запущених у ній додатків можна винести в окремий контейнер із власної проксі-функцією й окремим додатком буде запускатися у своєму власному мережному оточенні. У цьому підході, називаному контейнеризацією додатка, у контейнері розміщається тільки додаток, без операційної системи. У результаті щільність розміщення додатків буде майже в 100 разів більше, ніж при традиційної віртуалізації.

Контейнери – це перевірена технологія для досягнення високої щільності, великої гнучкості й необхідного масштабування. Вони можуть допомогти з упакуванням і масштабним розгортанням Web-Додатків, а використання їхніх властивостей в Web-Додатках і платформах сприяє рішенню деяких проблем, що виникають при наданні хмарних послуг, таких, наприклад, як багатокористувальницький доступ.

Однак контейнери – не універсальне рішення, тим більше що бізнес уже інвестував у такі орієнтовані на гіпервізори технології, як SR-IOV, VT-D і Network Function Virtualization. Більшість із них розроблені для здійснення прямого зв'язку між гостьовою віртуальною машиною й апаратною або комутаційною системою за допомогою спеціального драйвера, установленого в гостьовому ядрі. Тому що контейнерна технологія працює на рівні операційної

системи, вона не може «говорити» мовою «заліза», але майже для кожної корпоративної апаратної гіпервізорній технології (наприклад, SR-IOV або NFV) є рішення, що може бути використане в контейнері.

Однак реальність така, що контейнерний підхід не буде працювати, якщо міркувати про нього в парадигмі гіпервізора. Замість цього варто подумати, як зробити одну зі згаданих вище технологій доступної для контейнерів. Або просто оцінити потреби свого бізнесу й сучасні можливості віртуалізації й запитати себе, а що буде, якщо вибрати контейнери.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Віртуалізація – це, безумовно, революційна технологія. У даній роботі застосовується ефективний підхід до віртуалізації апаратних ресурсів процесора Cell Broadband Engine, що називається контейнерної віртуалізацією (або віртуалізацією операційної системи). Обговоримо елементи, необхідні для реалізації віртуалізації процесора Cell/B.E. програмними методами.

Поняття OpenVZ і процесора Cell/B.E.

OpenVZ – це програмна реалізація підтримки контейнерної віртуалізації в Linux з відкритим вихідним кодом. OpenVZ складається із двох компонентів:

– Ядро Linux OpenVZ.

– Інструменти простору користувача OpenVZ.

Основна мета проекту OpenVZ складається в створенні ізольованих контейнерів, які працюють під керуванням віртуальних екземплярів операційної системи Linux. У всіх контейнерів є доступ до єдиного віртуалізованому ядру. Базова система, що реалізує це ядро, обслуговує так звану основну четвірку віртуалізованих пристроїв: центральний процесор, оперативну пам'ять, дискові накопичувачі й мережні пристрої. Також можна передати один із пристроїв

прямо контейнеру, після чого воно стає недоступним базовій системі й всьому іншому контейнеру, за винятком контейнера, що володіє цим пристроєм (виділення пристрою в монопольне володіння).

Ви вже, імовірно, знайомі із процесором Cell/B.E., що застосовується в різних продуктах сторонніх виробників – від гібридних суперкомп'ютерів до спеціалізованих високозахищених апаратних прискорювачів і ігрової приставки Sony Playstation 3. Процесор Cell містить у собі ядро загального призначення з архітектурою Power Architecture™ і оптимізовані сопроцесорні елементи (streamlined co-processing elements, SPU), які значно прискорюють мультимедійні й векторні додатки. Процесор Cell/B.E. має продуктивність операцій із плаваючою точкою одиничної точності в 25,6 гігафлопс на SPU, за що його часто називають суперкомп'ютером у мікросхемі.

Базова архітектура складається із процесорного елемента Power (Power Processing Element, PPE), що являє собою ядро PowerPC зі звичайною підсистемою пам'яті, і восьми синергічних процесорних елементів (Synergistic Processing Elements, SPE), з'єднаних високошвидкісним внутрішнім каналом, називаним шиною з'єднання елементів (Element Interconnect Bus, EIB). Кожний SPE складається із синергічного процесора (Synergistic Processing Unit, SPU), локальної пам'яті (LS) обсягом 256 КБ, у якій зберігаються інструкції й дані SPE, а також контролера потоків пам'яті (MFC), що управляє передачею даних по DMA між основною пам'яттю системи й LS елемента SPE. PPE служить винятково для потреб операційної системи. На сьогоднішній день Linux є єдиною операційною системою, що підтримує архітектуру Cell/B.E.

Контейнерна віртуалізація й процесор Cell/B.E

Контейнерам надається доступ до виділеним фізичних SPU, наявним у системі. SPU, доступні в контейнері, недоступні в інших контейнерах і в базовій системі на час виділення. Кожний контейнер використовує тільки ті SPU, якими він володіє. Щоб реалізувати це, необхідно виконати чотири дії:

– Віртуалізувати файлову систему SPU (spuf). spuf повинна бути доступна усередині контейнерів, але кожний контейнер повинен бачити тільки потоки SPE, створені їм самим.

– Скорегувати віртуальну файлову систему, реалізовану в ядрі Linux 2.6 (sysfs). В /sys/devices/system/spu в sysfs усередині контейнера повинні втримуватися правильні записи каталогів для кожного SPU, виділеного йому. libspe2 використовує ці записи каталогів для підрахунку SPU, доступних усередині контейнера.

– Змінити планувальник SPU. Необхідно змінити планування SPU таким чином, щоб потоки SPE, створені усередині контейнера, працювали тільки на SPU, виділених цьому контейнеру.

– Змінити інструменти OpenVZ. Інструмент vzctl з пакета інструментів OpenVZ повинен підтримувати виділення SPU контейнерам і навпаки, звільняти SPU від контейнерів. Виділення й звільнення повинні функціонувати під час роботи контейнера.

Нам потрібний механізм, що буде контролювати, якою частиною пристрою володіє контейнер або протягом якого часу весь пристрій буде доступно контейнеру. Оскільки SPU неможливо розділити на частині фізично, контейнери використовують SPU з поділом за часом доступу.

У рамках нашої реалізації повинні виконуватися наступні чотири дії:

– Віртуалізувати spufs. spufs повинна бути доступна усередині контейнерів, але кожний контейнер повинен бачити тільки потоки SPE, створені їм самим.

– Скорегувати sysfs. У папці /sys/devices/system/spu в sysfs усередині контейнера повинні втримуватися правильні записи каталогів. Кращим рішенням буде наявність контейнерів, у яких є доступ до всім SPU, установленим у системі, тому в директорії /sys/devices/system/spu будуть однакові записи як на базовій системі, так і у всіх контейнерах.

– Змінити планувальник SPU. Планувальник SPU необхідно змінити таким чином, щоб потоки SPE, створені усередині контейнера, мали доступ тільки до SPU, доступним у системі на певний момент часу. Для потоків SPE може бути розроблений алгоритм планування, схожий із дворівневим рівномірним планувальником центрального процесора, що реалізувала команда OpenVZ для звичайних процесів. Можна впровадити навіть повністю новий алгоритм планування.

– Змінити інструменти OpenVZ. Інструмент vzctl з пакета інструментів OpenVZ повинен підтримувати установку часу виконання SPU для кожного контейнера. Цей параметр повинен бути змінюємо під час роботи контейнера.

Тепер небагато про spufs.

Використання spufs

Spufs сумісна з інший добре відомою віртуальною файловою системою (VFS) procfs. Procfs була першою абстрактною файловою системою в Linux, що була покликана максимально спростити подання процесів. У той час як procfs представляє процеси, що працюють на центральному процесорі, spufs представляє потоки, що працюють на SPU процесора Cell/B.E.

Spufs є спеціальною файловою системою. Це віртуальна файлова система розроблена для керування SPU на процесорі Cell/B.E. Кожна директорія spufs позначає логічний контекст SPE. Такий логічний контекст SPE розглядається так само, як фізичний SPE. Властивості контексту представлені у вигляді файлів у каталозі. При доступі до контексту SPE відбувається маніпуляція або реальним SPE, або його станом, збереженим в оперативній пам'яті. Для запуску програми SPU скопіюйте виконується файл, ЩО, ELF SPU в LS SPE і виконаєте системний виклик `spu_run`.

Якщо ви хочете виконати в Linux код, спеціально призначений для Cell/B.E., наприклад, програму, що працює на SPU процесора Cell/B.E., вам доведеться використовувати spufs. Іншого способу запуску на Linux процесів, що використовують SPU процесора Cell/B.E, не існує.

Як ви можете побачити в лістингу 1, перед виконанням команди `ls` на двох потоках SPE був запущений приклад програми швидкого перетворення Фур'є з Cell SDK. Точка монтування spufs – /spu.

Лістинг 1. Виконання приклада швидкого перетворення Фур'є з Cell SDK із двома потоками SPE

```
[root@c02b 12-0 ~]# ls /spu
spethread-20914-25296904 spethread-20914-25297464

[root@c02b 12-0 ~]# ls /spu/*
/spu/ spethread-20914-25296904:
cntl decr_status event_mask fpcr ibox_info lslr mbox_info mem mss
object-id proxydma_info regs signal1_type signal2_type wbox wbox_stat
decr dma_info event_status ibox ibox_stat mbox mbox_stat mfc npc physid
psmap signal1 signal2 srr0 wbox_info

/spu/ spethread-20914-25297464:
cntl decr_status event_mask fpcr ibox_info lslr mbox_info mem mss
object-id proxydma_info regs signal1_type signal2_type wbox wbox_stat
decr dma_info event_status ibox ibox_stat mbox mbox_stat mfc npc physid
psmap signal1 signal2 srr0 wbox_info
```

Результат роботи першої команди показує, що для кожного потоку SPE визначається одна директорія (kobject), у назві якої вказується ідентифікатор процесу (PID) і ідентифікатор потоку SPE. Результат роботи другої команди показує файли, що втримуються в обох директоріях.

Файл, що виконується, SPE у вигляді об'єктного файлу SPE прив’язується до потоку SPE, що виконується на фізичному SPE. Спеціальний планувальник SPE вирішує, коли і який потік SPE буде виконуватися на фізичному SPE. Об'єктні файли PPE обробляються так само, як і в системах стандартної архітектури PowerPC®, наприклад, на Blade-серверах JS21. Об'єктні файли PPE прив’язуються до стандартних потоків Linux, за керування якими відповідає планувальник Linux, реалізований у ядрі. Тому нічого нового в додатках, що працюють на PPE, немає.

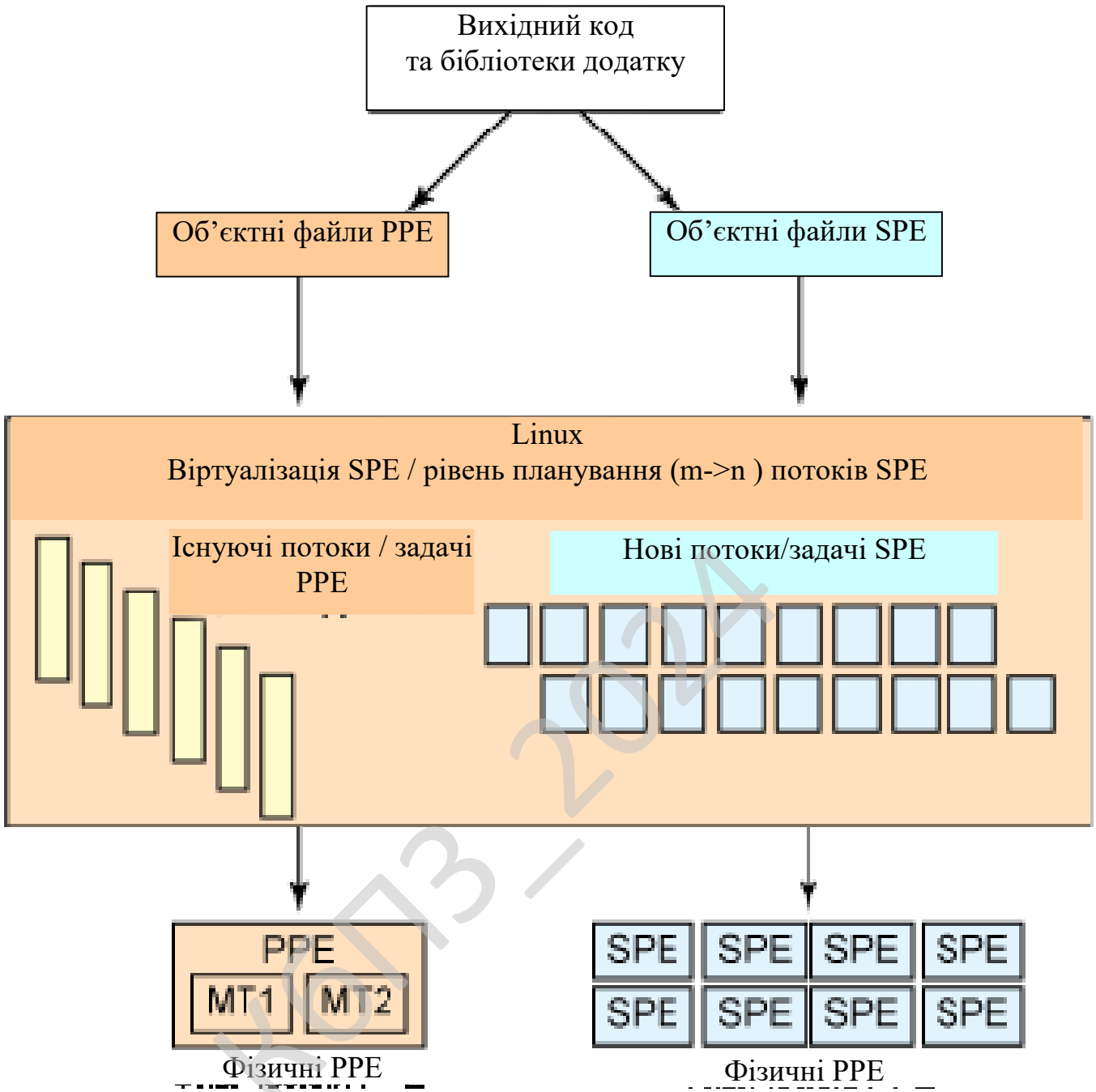

Рисунок 3.2 – Функціональна схема системи

Опис SDK Cell/B.E. і libspe

На сьогоднішній день випущена версія 3.0 комплекту для розробки програмного забезпечення (SDK) для Cell/B.E. (останню версію можна одержати в розділі завантаження центра ресурсів Cell). В SDK Cell/B.E. входять всі інструменти, необхідні для розробки додатків, що використовують процесори

Cell/B.E. Крім усього іншого, в SDK входять різні пакети компіляторів і компонувальників, бібліотеки, що спрощують розробку (наприклад, libspe), а також приклади програм SPU, MFC, LS і т.д., які демонструють використання можливостей архітектури Cell/B.E.

На сьогоднішній день libspe (версій libspe1 і libspe2) є єдиним користувачем spufs. Це середовище, що спрощує розробку коду для SPE архітектури Cell/B.E. Вона копіює виконуються файли, ЩО, ELF у локальну пам'ять SPE і виконує системний виклик `spu_run`, тому вам не потрібно піклуватися про внутрішні процеси запуску файлів, що виконуються, SPE.

На рисунку 6 показана ієрархія розширень, реалізованих в Linux для використання процесора Cell/B.E.

На рисунку показана бібліотека керування SPE у реальному часі (libspe), що працює в просторі користувача, що використовує файлову систему spufs, реалізовану в просторі ядра.

Опис sysfs

Споконвічно sysfs була реалізована в ядрі Linux як driverfs, її метою було надання загального огляду всіх пристроїв і драйверів, відомих ядру. Вона була розроблена як більше простій, чим procfs, спосіб доступу до пристроїв і драйверам. sysfs показує ієрархію структур даних kobject (представляючи кожну директорією) і безліч атрибутів (структури kobject), які є файлами, що звичайно містять одне значення, закодоване в текстовий рядок.

Наприклад, у лістингу 2 наведений лістинг директорії /sys/devices/system/spu на blade-сервері QS21 Cell/B.E.

Listing 2. One directory for each SPU

```
[root@c02b 12-0 ~]# ls /sys/devices/system/spu/
spu0  spu1  spu10  spu11  spu12  spu13  spu14  spu15  spu2  spu3  spu4
spu5  spu6
spu7  spu8  spu9
[root@c02b 12-0 ~]#
```

В blade-сервері QS21 Cell/B.E. використовується два процесори Cell/B.E. с вісьма SPU у кожному, таким чином, усього в сервері є 16 SPU. Як ви можете

бачити, у директорії /sys/devices/system/spu утримується по одній директорії для кожного SPU. libspe2 по цих записах директорій підраховує кількість фізичних SPU, доступних у системі.

Опис ядра OpenVZ

Ядро OpenVZ – це модифіковане ядро Linux, у якому реалізована можливість створення середовищ ізольованих контейнерів операційної системи. Крім того, воно виконує керування ресурсами й копіювання пам'яті в контрольних точках для контейнерів. Всі контейнери й навіть базова система використовують те саме загальне віртуалізоване ядро.

У кожному контейнері є власна безліч ресурсів, надаваних ядром, у число яких входять:

– Файли: системні бібліотеки й додатки.

– Віртуалізовані файлові системи: procfs або sysfs.

– Користувачі й групи: у кожному контейнері є власний користувач root, так само як і інших користувачах і групи.

– Дерево процесів: з контейнера видно тільки власна безліч процесів з віртуалізованими ідентифікаторами (PID процесу init дорівнює 1).

– Мережа: віртуальні мережні пристрої із власними адресами IP і правилами фільтрації й маршрутизації.

– Пристрою: віртуалізуються деякі пристрої. При необхідності будь-якому контейнеру може бути наданий доступ до реального (не віртуалізованого) пристрою.

– Об'єкти IPC: семафори, повідомлення й загальна пам'ять.

Керування ресурсами

Керування ресурсами виконується по різних типах ресурсів:

– Дискова квота: В OpenVZ реалізується дворівневе дискове квотування, що дозволяє обмежити дисковий простір для контейнера, а також дає контейнеру можливість установлювати квоти у власному середовищі.

– Планувальник центрального процесора: Рівномірний планувальник центрального процесора також використовує дворівневий механізм. Перший рівень цього планувальника, що набудовується – рівень контейнерів, на якому ви можете визначити, скільки процесорного часу буде виділено тому або іншому контейнеру. На другому рівні планувальник працює із плануванням процесів у середовищі усередині контейнерів.

– Лічильники beancounters користувачів: набір лічильників, обмежень і гарантій для ресурсів контейнерів. Існує порядку 20 параметрів пам'яті й різних об'єктів ядра, наприклад, сегментів загальної пам'яті IPC і мережних буферів.

Копіювання пам'яті в контрольних точках

Копіювання пам'яті в контрольних точках (checkpointing) – ще одна важлива функція системи. Копіювання й відновлення пам'яті в контрольних точках необхідно для міграції в реальному часі. Копіювання пам'яті в контрольних точках – це процес заморожування контейнера й наступного повного збереження його стану в дисковий файл. Відновлення являє собою зворотний процес. Міграція контейнера в реальному часі – це процес копіювання пам'яті контейнера в контрольній точці на одній базовій системі і її відновлення на іншій базовій системі.

Міграція в реальному часі – це одна дія, тоді як копіювання й відновлення пам'яті в контрольних точках – це дві різних дії, для виконання яких потрібне додатковий пристрій зберігання, доступне з обох апаратних вузлів.

Щоб продемонструвати реалізацію системи, у цій статті будуть освітлені наступні питання:

– Віртуалізація spufs: запобігання використання одного root-inode для всіх точок монтування.

– Коректування sysfs: запису необхідно коректувати в реальному часі.

– Модифікація планувальника SPU: додавання кроку реалізації віртуалізації.

– Модифікація інструментів OpenVZ: використання нових функцій.

Віртуалізація spufs

За замовчуванням spufs використовує той самий root-inode при кожному монтуванні. Наприклад, у вас може бути два середовища chroot (A й B), і spufs обох буде змонтована по шляху /spu. Якщо ви створите потоки SPE у середовищі A, після чого виведете вміст /spu, ви побачите тільки що створені потоки SPE. Однак у середовищі B ви побачите той же лістинг /spu, оскільки ви звертаєтеся до того же root-inode, що й у середовищі A.

Щоб змінити таке поводження, тобто щоб в обох середовищах можна було побачити тільки створені в них потоки SPE, необхідно змінити той факт, що spufs завжди використовує той самий root-inode для всіх точок монтування. При монтуванні spufs вона викликає функцію ядра, що повертає системний блок. Звичайно використовується функція get_sb_single(), що завжди повертає той самий системний блок. Функція get_sb_nodev() завжди повертає різні системні блоки, що приводить до потрібному нам результату.

OpenVZ дозволяє змонтувати усередині контейнера тільки кілька файлових систем. Кожна система реалізує екземпляр struct file_system_type , що визначає аспекти, пов'язані з файлами. OpenVZ розширює цю структуру (поряд з іншими), доповнюючи її компонентом, що визначає, чи може бути змонтована файлова система. Таким чином, компоненту .fs_flags повинно бути встановлене значення FS_VIRTUALIZED.

У процесу, створеного усередині контейнера, є два PID. У кожного контейнера є власна безліч ресурсів, надаваних ядром. У дерева процесів була одна безліч ресурсів, а контейнер міг бачити тільки власну безліч процесів з віртуалізованими PID. Усередині контейнера є віртуалізований PID, а в базовій системі той же процес ідентифікується іншим PID, що називається глобальним PID. Як уже згадувалося, назва директорій spufs задається у вигляді spethread-<PID>-< thread-ID>, де <PID> – ідентифікатор процесу потоку SPE, а < thread-ID>> відповідний ідентифікатор потоку. Оскільки в середовищі контейнера використовуються віртуалізовані PID, лістинг spufs

усередині контейнера повинен показувати віртуалізовані, а не глобальні PID. На щастя, такий алгоритм уже реалізований в OpenVZ.

Зміните файл arch/powerpc/platforms/cell/spufs/inode.c у ядрі Linux®.

Коректування sysfs

Sysfs уже віртуалізована, тому її можна використовувати усередині контейнерів. Вона є не копією sysfs, видимої в базовій системі, а її частиною. Ціль полягає в тому, щоб виділяти SPU контейнерам і звільняти їх у реальному часі. Це означає, що запису sysfs також необхідно адаптувати в реальному часі. Перед цим створіть директорію, у якій перераховані SPU.

Це директорія /sys/devices/system/spu. За замовчуванням в sysfs у середовищі контейнера немає директорій /sys/devices/system/spu, /sys/devices/system і /sys/devices. Ці три записи необхідно створити під час ініціалізації (запуску) контейнера. Піддиректорії /sys/devices/system/spu можуть мати вигляд від spu0 до spu<N>, де <N> – кількість доступних SPU у системі мінус 1. Директорії необхідно створювати, коли SPU призначається контейнеру, і видаляти, коли вони відкріплюються від контейнера.

Кожної директорії в лістингу sysfs повинен відповідати екземпляр kobject у просторі ядра. kobject – це структура, що визначає назву директорії і її батьківський kobject (інакше кажучи, що відповідає батьківську директорію). Наприклад, в kobject, що відображається як директорія /sys/devices/system/spu/spu3, є два параметри, які потрібно змінити:

– Назва spu3.

– Батько (покажчик на kobject, що представляє /sys/devices/system/spu).

Після реєстрації kobject в sysfs за допомогою виклику `subsystem_register()`, він стає видимим із простору користувача. Зворотна дія складається у видаленні директорії з sysfs. Для цього можна викликати функцію `subsystem_unregister()`, указавши kobject, якому необхідно видалити.

В kobject з назвою spu3 є покажчик на батьківський kobject з назвою spu.

Основні зміни необхідно зробити у файлі ядра kernel/ve/vecalls.c. Це файл OpenVZ, у якому реалізована більшість функцій, які викликаються під час ініціалізації й настроювання параметрів контейнерів.

Модифікація планувальника SPU

Кожний фізичний SPU системи представлений екземпляром структури `spu` у просторі ядра. У цій структурі зберігається кілька параметрів, у число яких входять:

– Ідентифікатор (номер) SPU.

– До якого вузла Cell/B.E. він належить.

– Покажчик на LS SPU.

Нова змінна зберігає власника SPU у вигляді ідентифікатора контейнера. Планувальник SPU реалізує функцію `spu_alloc()`, що шукає вільний SPU, на якому буде виконуватися потік SPE. Таким чином, він шукає список доступних у системі SPU (на які в цей момент не виконується потоків SPE).

Звичайно береться перший SPU у списку й на ньому запускається потік SPE. Щоб реалізувати віртуалізацію, функція повинна перевірити, чи збігається ідентифікатор контейнера вільного SPU з ідентифікатором контейнера потоку SPE, що передбачається на ньому виконати.

Якщо ця додаткова перевірка дає негативний результат, функція перевіряє наступний елемент у списку вільних SPU. Якщо вільних SPU для контейнера, що запускає потік SPE, ні, планувальник SPU поводиться так само, як якби список був порожній, і чекає звільнення SPU.

Функція `spu_alloc()` реалізована у вихідному файлі ядра Linux arch/powerpc/platforms/cell/spu_base.c.

Модифікація інструментів OpenVZ

Більша частина потрібної функціональності вже реалізована, але для того, щоб використовувати нові функції, необхідно модифікувати інструменти OpenVZ. Інструмент vzctl управляє виділенням SPU у реальному часі. Це

основний інструмент для настроювання параметрів в OpenVZ. Новий параметр для установки кількості SPU, виділених контейнеру, – `---spus <nr_spus>`.

Значення `<nr_spus>` представляє кількість SPU, виділених контейнеру. Це абсолютне значення, тому якщо контейнеру з вісьма SPU привласнюється значення 6, то від процесора відкріплюються 2 SPU, а не додаються 6 SPU (8 – 6 = 2).

Наприклад, от результат роботи команди, коли контейнер з ID, рівним 101, одержує вісім SPU:

```
[root@c02b 12-0 ~]# vzctl set 101 ---ispus 8
Setting SPUs: 8
Configure meminfo: 1024000
WARNING: Settings were not saved and will be reset to original
 values on next start (use ---isave flag)
[root@c02b 12-0 ~]#
```

Для реалізації такого алгоритму інструмент vzctl повинен перейти бар'єр простору користувача й виконати ряд операцій керування в просторі ядра. Інструмент повинен знайти SPU, які не використовуються іншими контейнерами. Інструмент vzctl виконує пошук за списком доступних SPU і перевіряє ідентифікатор контейнера в структурі нового spu (описана в розділі модифікації планувальника SPU). Якщо це значення дорівнює 0, SPU може бути призначений розглянутому контейнеру. Значення 0 використовується тому, що значення ідентифікатора контейнера повинне бути більше 0, тобто значення 0 указує на те, що SPU не призначено жодному з контейнерів. Якщо функція не може знайти потрібного для виконання запиту кількості вільних SPU, процедура завершується й не призначає контейнеру жодного SPU. Якщо кількість SPU, уже призначених контейнеру, більше запитаного кількості SPU, різниця відкріплюється.

Щоб перебороти бар'єр між простором користувача й простором ядра, можна використовувати різні моделі реалізації. Найбільш простий спосіб складається в реалізації нового системного виклику, що накладає параметри `<containerID>` і `<nr_spus>` на параметри системного виклику.

Функції, які виконують настроювання параметрів SPU контейнера, повинні реалізовуватися в тій частині ядра, що може бути реалізована в модулі ядра. Це представляє більшу проблему. Якщо модуль ядра не завантажений, функція обробки системного виклику в просторі ядра нічого не зробить. Однак якщо модуль завантажений, він викличе функцію, реалізовану в модулі. Це нетривіальне завдання, оскільки таблиця системних викликів (де зберігаються покажчики функції на функції обробки системних викликів) є частиною статичного складання ядра.

Модуль не є частиною цієї статичної функції, і тому статична убудована функція обробки системного виклику не може викликати функцію, що є частиною модуля. Рішення складається в реалізації оболонки функції, що копіює покажчик на функції модуля в змінну убудованій статичній функції обробки системного виклику таким чином, щоб убудований статичний оброблювач системного виклику міг викликати функцію модуля. Ця оболонка функції викликається під час ініціалізації й очищення модуля.

Ви бачите, як покажчик функції, реалізованої в модулі, копіюється в убудовану статичну функцію простору ядра. Пунктирні стрілки показують, як додатка простору користувача викликають функцію модуля, передаючи убудовану статичну функцію оброблювача системного виклику.

Необхідно змінити вихідний код ядра в наступних файлах:

– include/ asm-powerpc/systbl.h

– include/ asm-powerpc/unistd.h

– include/linux/syscalls.h

– kernel/sys.c

– kernel/sys_ni.c

– kernel/ve/vecalls.c

Крім того, обновляються файли вихідного коду vzctl OpenVZ:

– include/res.h

– include/vzctl_param.h

– include/vzsyscalls.h

– src/lib/config.c

– src/lib/res.c

– src/vzctl.c

У систему складання OpenVZ додається два нових файли:

– include/spu.h

– src/lib/spu.c

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

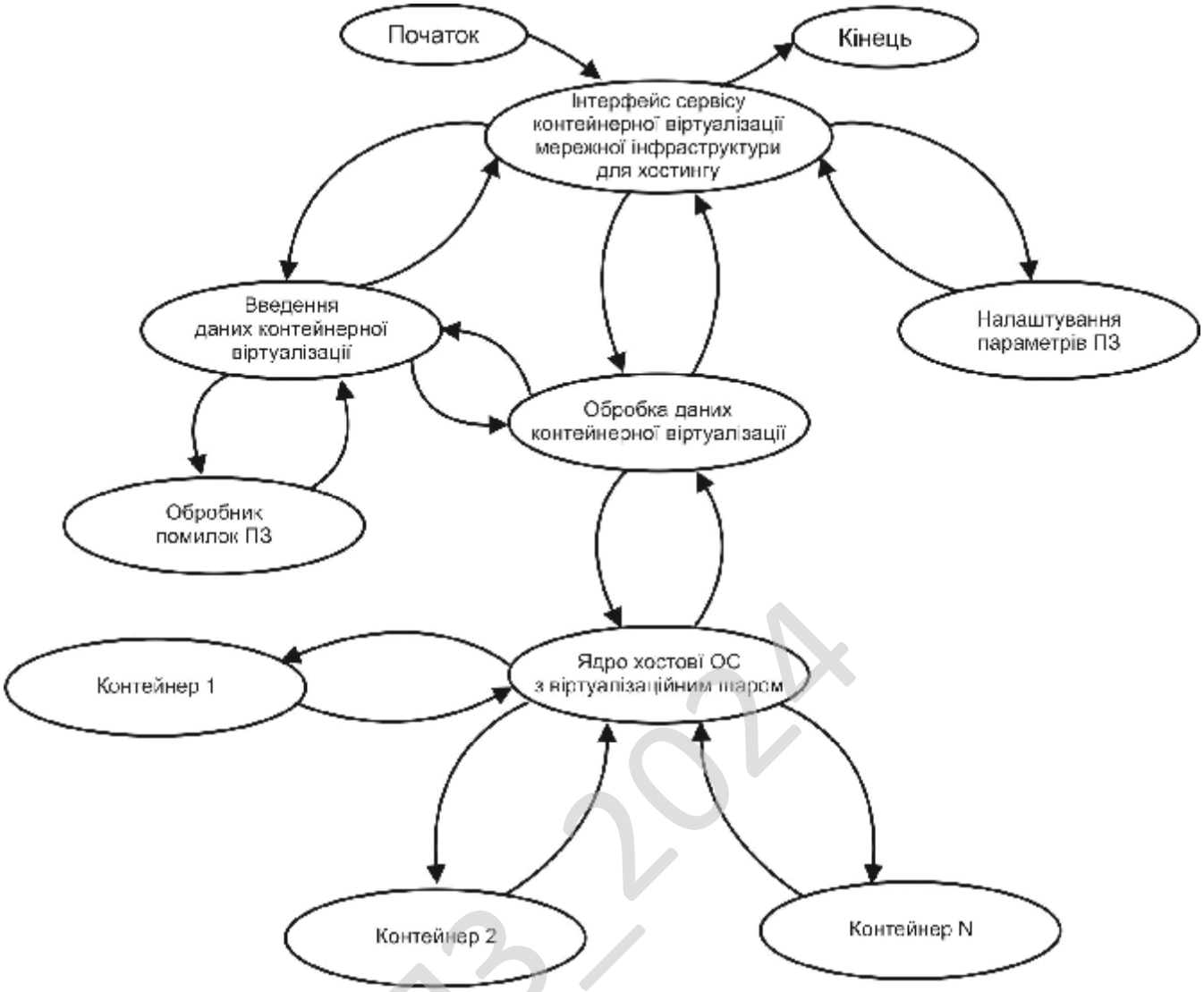

Рисунок 3.3 – Діаграма взаємодії процесів

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

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю сервісу контейнерної віртуалізації мережної інфраструктури для хостингу.

Під час роботи над бакалаврською дипломною роботою 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.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

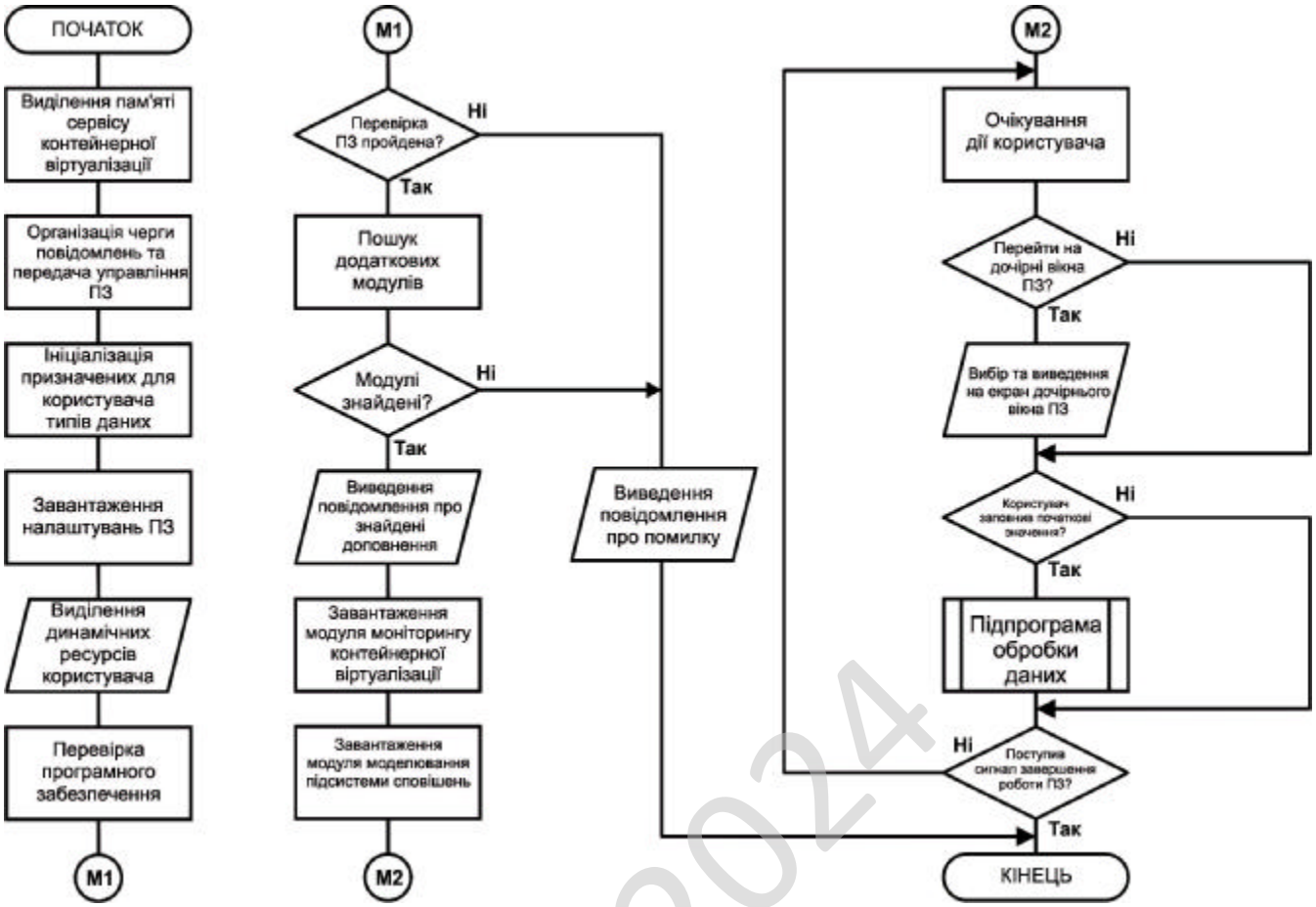

Рисунок 4.1 – Блок-схема основної програми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML.

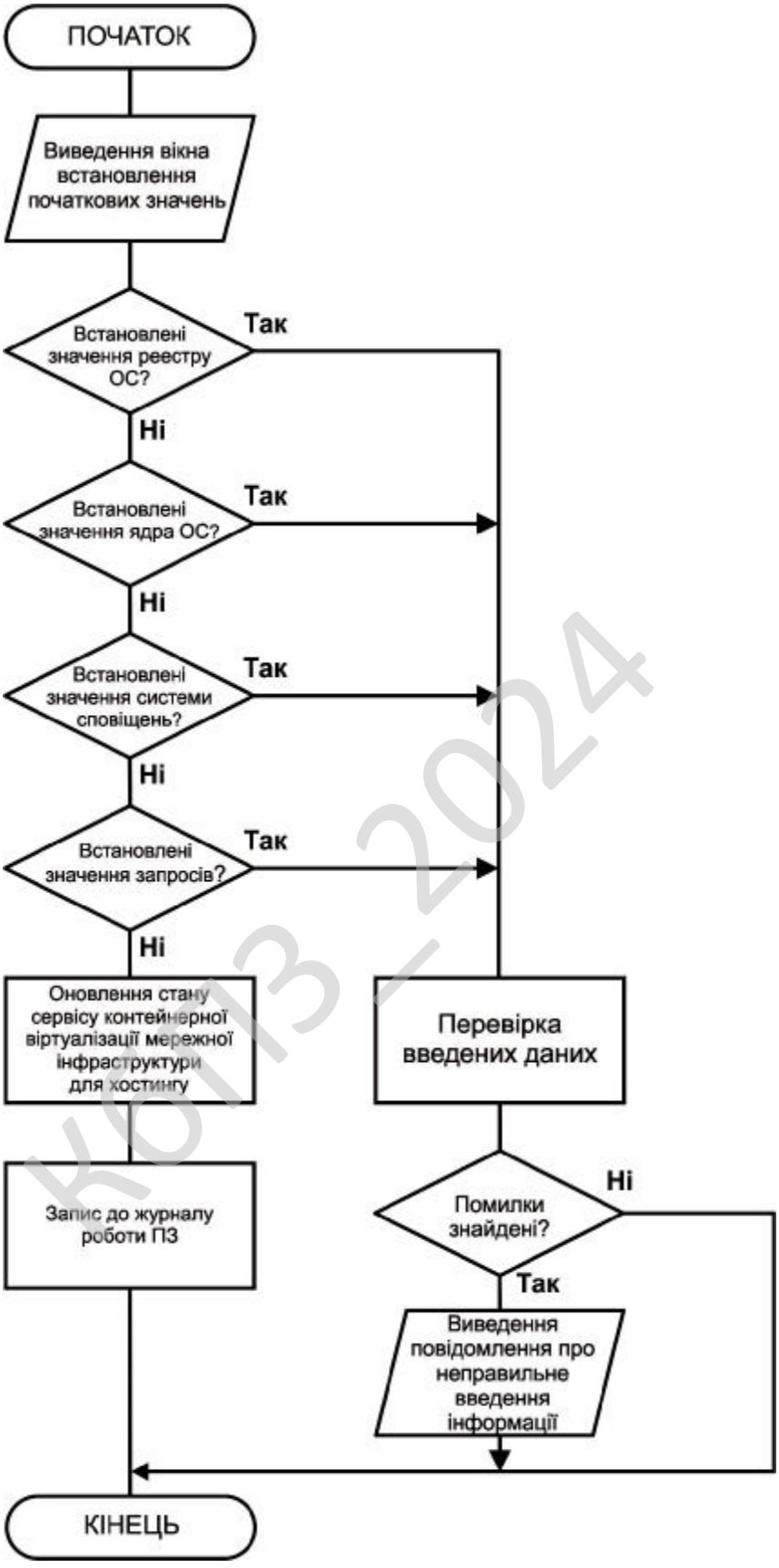

Рисунок 4.2 – Блок-схема роботи підпрограми

Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: Діаграма класів; Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого

програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку

можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу.

Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій

відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду.

Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент.

Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

У роботі використовувались дві мови програмування Builder C++ та Delphi XE8. Що дало змогу підвищити надійність та стабільність реалізації сервісу контейнерної віртуалізації мережної інфраструктури для хостингу. Одне з першечергових завдань віртуалізації мережної інфраструктури – забезпечення системи надійним алгоритмом шифрування внутрішніх модифікованих даних.

Методи захисту даних на персональних комп’ютерах дуже різноманітні як по кінцевій цілі, так і по технічному втіленню.

Нижче наведено розроблений вихідний код, методу шифрування внутрішніх модифікованих даних.

```
(*
Схема дії алгоритму
1. У функцію передається масив даних.
2. Генерується унікальний 128 бітний ключ для кодування на основі guid.
3. Обчислюється розмір даних, вони повинні бути кратні парній кіл-ті 32 бітних
блоків, об’єм, що бракує заповнюється випадковою інформацією.
4. Перші три байти займає заголовок eta
5. У четверту позицію даних додається значення яке указує обсяг сміття.
6. Виробляється кодування раніше згенерованим ключем. Сам ключ кодується
внутрішнім ключем, ділиться на чотири частини.
7. 128 бітний ключ розділяється на чотири 32 бітних блоки
8. Ключ додається в дані зі зрушенням у будь-яке місце
в наступному порядку 4-ий, 3-ий, 1-ий, 2-ий блоки.
9. На 14 у позицію даних вставляється зі зрушенням покажчик на початок ключа.
10. Разом у закодованому буфері перебуває крім основної ще й 21 байт службової
інформації. Для раскодування всі дії виробляється у зворотному порядку
*)
unit Model;
```

```
interface
uses
windows,
sysutils,
classes;
const
delta=$9e3779b9;
// Зсув контрольної суми ~ 32 біт
_k0: int64 = $982c98c1;
// Головний 128 бітний ключ (4 частини по 32 біта)
_k1: int64 = $7f8f4d4b;
 _k2: int64 = $bcae3151;
_k3: int64 = $971bc789;
header = ' Program_Model '; // Заголовок
type
phash = ^thash;
thash = array of byte;
function decript(var s: thash): boolean;
procedure encript(var s: thash);
implementation
////////////////////////////////////////////////////
// ЧАСТИНА ПЕРША * * * КОДУВАННЯ * * *
procedure encript(var s: thash);
var
inbuf,
outbuf,
resultbuf: thash;
// Вхідний, вихідний і результуючий буфери
y, z, sum: longword;
// Тимчасові змінні для кодуємих блоків даних
k0, k1, k2 , k3: longword;
// Поточний ключ для шифрування
i, a, len: integer;
// Змінні для циклів
c: byte;
// Лічильник кіл-ті сміття
guid, key: string;
g: tguid;
begin
// Перевірка розміру даних
if length(s) = 0 then exit;
createguid(g); // Генеруємо ключ на основі guid
```

```
guid := guidtostring(g);
for i := 1 to length(guid) do
if guid[i] in ['0'..'9', 'a'..'f'] then
key := key + guid[i];
k0 := strtoint64('$' + copy(key, 1, 8));
k1 := strtoint64('$' + copy(key, 9, 8));
k2 := strtoint64('$' + copy(key, 17, 8));
k3 := strtoint64('$' + copy(key, 25, 8));
c := 0; // Ініціалізуємо лічильник дозаповнень
// Дозаповняємо дані щоб останній блок даних був рівний 64 бітам
while (length(s) div 8) * 8 <> length(s) do
begin
len := length(s);
inc(len);
setlength(s, len);
s[len – 1] := random(255); // Заповнюємо випадковими даними
inc(c);
end;
len := length(s); // Обчислюємо розмір кодуємого блоку
setlength(inbuf, len); // Установлюємо розмір буферів
// Розмір вихідного буфера збільшений на 21 байт через
// 3 байти – заголовок eta
// 1 байт – лічильник кіл-ті сміття наприкінці буфера
// 16 байт – кодований ключ – 4 cardinal
// 1 байт – мітка розташування кодованого ключа
inc(len, 21);
setlength(outbuf, len);
setlength(resultbuf, len);
//Збільшимо кіл-у сміття для видалення самого поля лічильника
inc(c);
 outbuf[0] := ord(header[1]); // додаємо ідентифікатор
outbuf[1] := ord(header[2]);
outbuf[2] := ord(header[3]);
outbuf[3] := c; // Додаємо лічильник сміття
move(s[0], inbuf[0], length(s));
// Заповнюємо вхідний буфер даними
i := 0;
while i < len – 21 do // Безпосередньо кодування
begin
move(inbuf[i], y, 4); // Беремо перші 32 біта
move(inbuf[i + 4], z, 4); // Беремо другі 32 біта
// Кодуємо
```

```
sum := 0;
// 64 бітний кодуємий блок (2 частини по 32 біта)
for a := 0 to 31 do
begin
inc(sum, delta);
inc(y,((z shl 4)+k0)xor(z+sum)xor((z shr 5) + k1));
inc(z,((y shl 4)+k2) xor (y+sum) xor ((y shr 5)+k3));
end;
move (y, outbuf[i + 4], 4);
// Поміщаємо кодовані блоки у вихідному буфер
move (z, outbuf[i + 8], 4);
inc(i, 8);
// Пропускаємо оброблений блок, переходимо до наступного
end;
sum := 0;
for a := 0 to 31 do
// Кодуємо перші 2 частини ключа внутрішнім ключем
begin
inc(sum,delta);
inc(k0,((k1 shl 4)+_k0)xor(k1+sum)xor((k1 shr 5)+_k1));
inc(k1,((k0 shl 4)+_k2)xor(k0+sum)xor((k0 shr 5)+_k3));
end;
sum := 0;
for a := 0 to 31 do
// Кодуємо другі 2 частини ключа внутрішнім ключем
begin
inc(sum, delta);
inc(k2,((k3 shl 4)+_k0) xor (k3+sum)xor((k3 shr 5)+_k1));
inc(k3,((k2 shl 4)+_k2) xor (k2+sum)xor((k2 shr 5)+_k3));
 end;
// Визначаємо позицію розміщення ключа в блоці даних
randomize;
if len < 255 then
i := len
else
i := 255;
repeat
i := random(i);
if i < 4 then i := 4;
until i <= len - 16;
// Зміщаємо дані звільняючи місце для чотирьох
// частин ключа
```

```
move(outbuf[0], resultbuf[0], i);
move(outbuf[i], resultbuf[i + 16], len - i - 17);
// Розбиваємо четверту чверть ключа на 4 восьмибітних //частини
resultbuf[i] := byte(k3 shr 24);
resultbuf[i + 1] := byte(k3 shr 16);
resultbuf[i + 2] := byte(k3 shr 8);
resultbuf[i + 3] := byte(k3);
// Розбиваємо третю чверть ключа на 4 восьмибітних частини
resultbuf[i + 4] := byte(k2 shr 24);
resultbuf[i + 5] := byte(k2 shr 16);
resultbuf[i + 6] := byte(k2 shr 8);
resultbuf[i + 7] := byte(k2);
// Розбиваємо першу чверть ключа на 4 восьмибітних частини
resultbuf[i + 8] := byte(k0 shr 24);
resultbuf[i + 9] := byte(k0 shr 16);
resultbuf[i + 10] := byte(k0 shr 8);
resultbuf[i + 11] := byte(k0);
// Розбиваємо другу чверть ключа на 4 восьмибітних частини
resultbuf[i + 12] := byte(k1 shr 24);
resultbuf[i + 13] := byte(k1 shr 16);
resultbuf[i + 14] := byte(k1 shr 8);
resultbuf[i + 15] := byte(k1);
// Зрушуємо дані з 14 позиції на одну вправо для мітки
// (буфер починається з нуля)
for a := len - 1 downto 14 do
resultbuf[a] := resultbuf[a - 1];
// Поміщаємо мітку початку ключа ( 14-й байт)
resultbuf[13] := i;
s := resultbuf;
end;
////////////////////////////////////////////////////
//ЧАСТИНА ДРУГА * * * ДЕКОДУВАННЯ * * *
function decript (var s: thash): boolean;
var
inbuf,
outbuf,
resultbuf: thash;
//Вхідний, вихідний і результуючий буфери
y , z, sum: longword;
// Тимчасові змінні для кодуємих блоків даних
k0, k1, k2, k3: longword; // Поточний ключ для шифрування
i, a, len: integer; // Змінні для циклів
```

```
aheader: string;
begin
result := false;
len := length(s); // Обчислюємо розмір декодуємого блоку
// Перевірка розміру
if len < 27 then exit;
if len <> (((len - 21) div 8) * 8) + 21 then exit;
// Перевірка заголовка
aheader := char(s[0]) + char(s[1]) + char(s[2]);
if aheader <> header then exit;
// Перевірка позиції ключа
if not(s[13] in [4..255]) then exit;
if s[13] + 16 > len then exit;
// Перевірка лічильника сміття
if s[3] > 8 then exit;
setlength(inbuf, len);
// Установлюємо розмір буферів
move(s[0], inbuf[0], len);
// Заповнюємо вхідний буфер даними
i := inbuf[13]; // Довідаємося початкову позицію ключа
// Видаляємо мітку на початок ключа
for a := 13 to len - 2 do
begin
inbuf[a] := inbuf[a + 1];
inbuf[a + 1] := 0;
end;
dec(len);
// Витягаємо ключ
k3:=(inbuf[i+3]or(inbuf[i+2] shl 8)or(inbuf[i+1] shl 16) or (inbuf[i] shl 24));
k2:=(inbuf[i+7]or(inbuf[i+6]shl 8)or(inbuf[i+5]shl 16)or (inbuf[i+4] shl 24));
k0:=(inbuf[i+11]or(inbuf[i+10]shl 8)or(inbuf[i+9]shl 16) or (inbuf[i+8] shl 24));
k1:=(inbuf[i+15]or(inbuf[i+14]shl 8)or(inbuf[i+13]shl 16) or (inbuf[i+12] shl
24));
// Видаляємо ключ із блоку даних
for a := i + 16 to len do
begin
inbuf[a - 16] := inbuf[a];
inbuf[a] := 0;
end;
setlength(outbuf, len);
zeromemory(outbuf, len);
dec(len, 16); // Видаляємо розмір ключа
```

```
// Декодуєм перші дві частини ключа
sum := delta shl 5;
for a := 0 to 31 do
begin
dec(k1,((k0 shl 4)+_k2)xor(k0+sum) xor ((k0 shr 5)+_k3));
dec(k0,((k1 shl 4)+_k0) xor (k1+sum)xor((k1 shr 5)+_k1));
dec(sum, delta);
end;

// Декодуєм другі дві частини ключа
sum := delta shl 5;
for a := 0 to 31 do
begin
dec(k3,((k2 shl 4)+_k2)xor(k2+sum)xor((k2 shr 5) + _k3));
dec(k2,((k3 shl 4)+_k0)xor(k3+sum) xor ((k3 shr 5)+_k1));
dec(sum, delta);
end;
i := 0;
dec(len); // Видаляємо з розміру місце лічильника сміття
dec(len, 3); // Видаляємо з розміру заголовок eta
while i < len do // Безпосередньо декодировка
begin
move(inbuf[i + 4], y, 4); // Беремо перші 32 біта
move(inbuf[i + 8], z, 4); // Беремо другі 32 біта
// Декодуєм
sum := delta shl 5;
for a := 0 to 31 do
// 64 бітний кодуємий блок (2 частини по 32 біта)
begin
dec(z,((y shl 4)+k2)xor(y+sum)xor((y shr 5)+k3));
dec(y,((z shl 4)+k0)xor(z+sum)xor((z shr 5)+ k1));
dec(sum, delta);
end;
move(y, outbuf[i], 4);
// Запам'ятовуємо кодовані блоки у вихідному буфері
move(z, outbuf[i + 4], 4);
inc(i, 8);
// Пропускаємо оброблений блок, переходимо до наступного
end;
//Відрізаємо -1 тому що місце для лічильника вже вилучене з //len
 len := len - (inbuf[3] - 1);
setlength(resultbuf, len);
```

```
move(outbuf[0], resultbuf[0], len);
// Виводимо текст із вихідного буфера
s := resultbuf;
result := true;
end;
end.
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм LOKI_91. Механізм алгоритму LOKI_91 подібний DES (рисунок 4.3). Блок даних розщеплюється на ліву й праву половини й проходить 16 раундів, що досить нагадує DES. У кожному раунді права половина спочатку піддається операції XOR із частиною ключа, а потім розширювальній перестановці (таблиця 4.1).

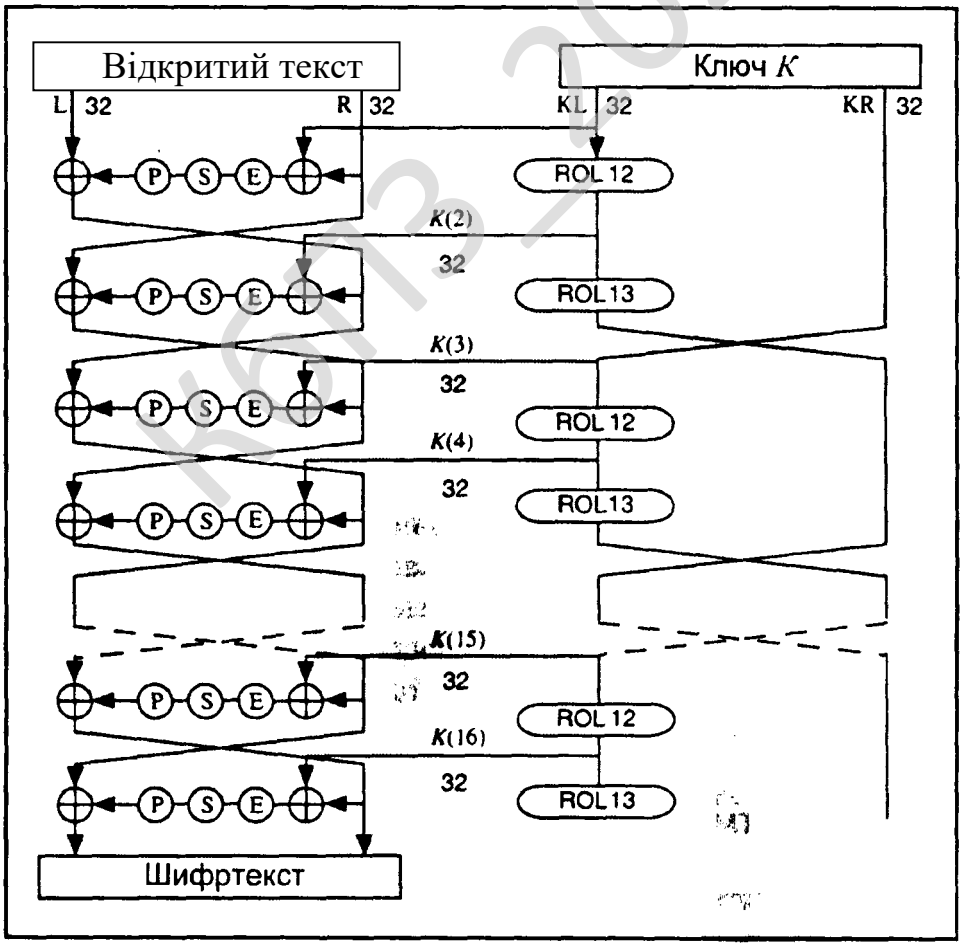

Рисунок 4.3 – Алгоритм LOKI91

Таблиця 4.1 – Перестановка з розширенням

4	3	2	1	32	31	30	29	28	27	26	25
28	27	26	25	24	23	22	21	20	19	18	17
20	19	18	17	16	15	14	13	12	11	10	9
12	11	10	9	8	7	6	5	4	3	2	1

48-бітовий вихід розділяється на чотири 12-бітових блоки. У кожному блоці виконується така підстановка з використанням S-блоку: береться кожний 12-бітовий вхід, 2 старших і 2 молодших біти використовуються для утворення номера r, а вісім внутрішніх біт утворять номер c. Вихід S-блоку, O, має наступне значення: $O(r,c) = (c + ((r*17) \oplus 0xff) \& 0xff)^{31} \bmod P_r$.

Таблиця 4.2 – Значення P_r

r	1	2	3	4	5	6	7	8
P_r	375	379	391	395	397	415	419	425
r	9	10	11	12	13	14	15	16
P_r	433	445	451	463	471	477	487	499

Після цього чотири 8-бітових результати знову поєднуються, утворюючи 32-бітове число, що піддається операції перестановки, описаній в таблиці 3. Нарешті, для одержання нової лівої половини виконується операція XOR правої половини з колишньою лівою половиною, а ліва половина стає новою правою половиною. Після 16 раундів для одержання остаточного шифртексту знову виконується операція XOR над блоком і ключем.

Таблиця 4.3 – Перестановка за допомогою P-блоку

32	24	16	8	31	23	15	7	30	22	14	6	29	21	13	5
2	20	12	4	27	19	11	3	26	18	10	2	25	17	9	1

Підключи генеруються із ключа досить прямолінійно. 64-бітовий ключ розбивається на ліву й праву половини. На кожному раунді підключем служить ліва половина. Далі вона циклічно зрушується вліво на 12 або 13 біт, потім після кожних двох раундів ліва й права половини міняються місцями. Як і в DES, для зашифрування й розшифрування використовується один й той самий алгоритм із деякими змінами у використанні підключів.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи. Розроблене програмне забезпечення сервісу контейнерної віртуалізації мережної інфраструктури для хостингу складається з наступних функціональних блоків:

– Навігаційне меню: довідкова інформація контейнеру; Контейнери; Виконання дії; Довідка.

– Блоку налаштування параметрів сервісу контейнерної віртуалізації мережної інфраструктури для хостингу.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

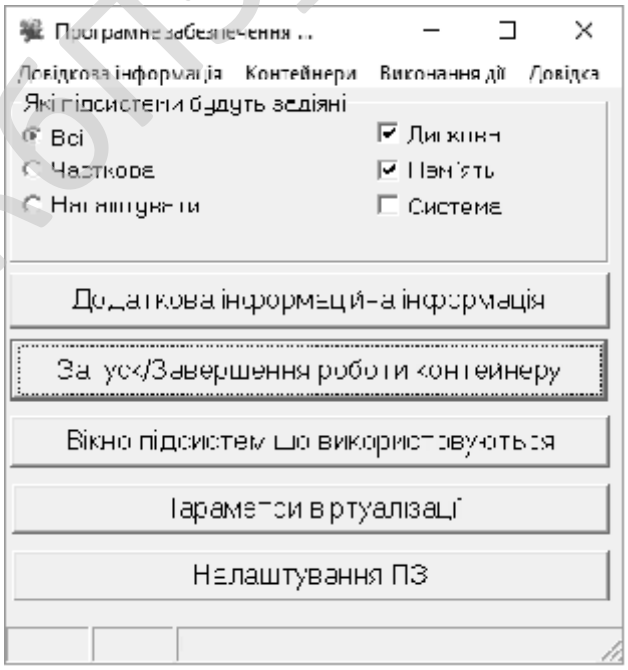

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

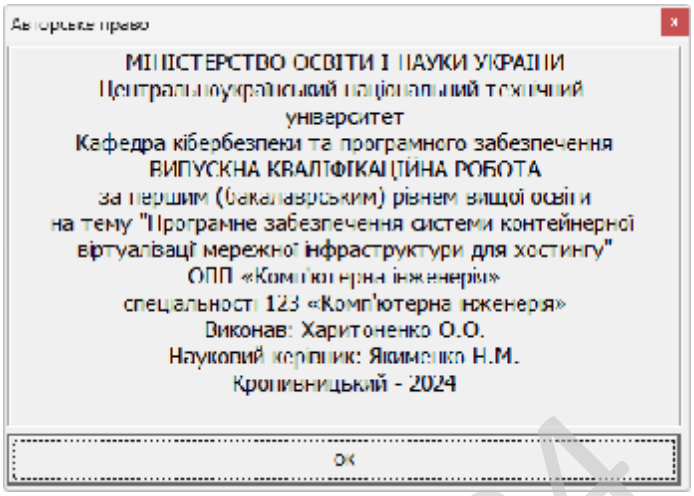

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки та чорної скриньки.

Тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТс;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно.

Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ).

Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи контейнерної віртуалізації мережної інфраструктури для хостингу.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем контейнерної віртуалізації мережної інфраструктури для хостингу.

– Досліджена система контейнерної віртуалізації мережної інфраструктури для хостингу.

– На основі отриманих результатів досліджень створена програмна реалізація системи контейнерної віртуалізації мережної інфраструктури для хостингу.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання контейнерної віртуалізації мережної інфраструктури для хостингу.

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

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

Програма реалізована на мові високого рівня RAD Studio Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи контейнерної віртуалізації мережної інфраструктури для хостингу. Це

дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм LOKI_91.

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. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

2. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

3. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

4. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

5. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

6. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

7. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

8. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

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

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

11. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

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

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

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

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

16. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties».

International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

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

18. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

19. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

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

21. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

22. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

23. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

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

25. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

26. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

27. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

28. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

29. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

30. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного

захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

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

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

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

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

35. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

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

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

38. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

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

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

41. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

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

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

44. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

45. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

46. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

47. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

48. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. - 2016. - С. 121-127.

49. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. - Випуск 2 (46) - Х.: ХУПС - 2016. - С. 146-149.

50. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

51. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

52. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи контейнерної віртуалізації мережної інфраструктури для хостингу.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 133-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи контейнерної віртуалізації мережної інфраструктури для хостингу.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи контейнерної віртуалізації мережної інфраструктури для хостингу;

– 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, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 74 аркуші.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 5.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Якименко Н.М.

Програмне забезпечення системи контейнерної віртуалізації мережної інфраструктури для хостингу

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 39

Літера: РП

Кропивницький – 2024 року

Основний файл проекту розробленого програмного забезпечення (Program_Model_LinuxConteinerVirtualization.dpr)

```
program ProgramModelLinuxConteinerVirtualization;

uses
  Forms,
  Messages,Graphics,
  Windows,Unit1 in 'Unit1.pas' {Form1},
  Unit2 in 'KHARYTONENKOReport.pas' {Form2},
  Unit3 in 'fset.pas' {Form3},
  Unit4 in 'data.pas' {Form4},
  Unit5 in 'LinuxConteinerVirtualizationEmulation.pas' {Form5},
  Unit6 in 'U6.pas' {Form6},
  U_splash in 'U7.pas' {U_Form_Splash}.

const
WM_CONST1=WM_USER+322; WM_CONST2=WM_USER+105; WM_CONST3=WM_USER+99;

{$R *.res}

///Розробив Харитоненко Олександр Олександрович, КМ-20, ЦНТУ, 2024

begin
  Application.HintPause:=400; Application.HintHidePause:=1000;
  Application.Title:= ProgramModelLinuxConteinerVirtualization';
  Application.Initialize;
try
    U_Form_Splash:=TU_Form_Splash.Create(Application);
    U_Form_Splash.Show;
    U_Form_Splash.Update;
  Sendmessage(U_Form_Splash.Handle,WM_MY,0,'Start');
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm4, Form4);
  Sendmessage(U_Form_Splash.Handle,WM_MY,0,'End');
  Application.CreateForm(TForm5, Form5);
Finally U_Form_Splash.free; end;
  Application.Run;
end.
```

Файл форми взаємодії та відображення звіту роботи
(KHARYTONENKOReport.pas)

```
unit KHARYTONENKOReport;

interface

uses
Windows, glReport, glCapt, glBevel, glPage, Printers,
glLabel, glRuler, Mask, glLBox, dsgnintf, Spin, geRPEdit,
Menus, ExtCtrls, StdCtrls, Buttons, ComCtrls, Controls, Dialogs,
Forms, Classes,Sysutils;
///Розробив Харитоненко Олександр Олександрович, КМ-20, ЦНТУ, 2024

type

tmyRep_Property = class( TPropertyEditor )
function GetAttributes: TPropertyAttributes; override;
function GetValue: string; override;
procedure Edit; override;
end;

tmyRep_Editor = class(TComponentEditor)
procedure ExecuteVerb(Index: Integer); override;
function GetVerb(Index: Integer): string; override;
function GetVerbCount: Integer; override;
end;

tmyKHARYTONENKOReportEditor = class(TComponent)
FKHARYTONENKOReport: tmyKHARYTONENKOReport;
protected
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
public
procedure Preview;
procedure Edit;
published
property KHARYTONENKOReport: tmyKHARYTONENKOReport read FKHARYTONENKOReport
write FKHARYTONENKOReport;
end;

tmyRepEditor = class(TForm) //основний классвзаємодії
OpenDialog1: TOpenDialog; SaveDialog1: TSaveDialog;
PM_Control: TPopupMenu; N_Linktofile: TMenuItem;
N1: TMenuItem; N2: TMenuItem;
PC: tmyPageControl; Panel1: TPanel;
Bevel4: TBevel; P_Sides: TPanel;
Panel2: TPanel; glBevel1: tmyBevel;
Bevel2: TBevel; Bevel1: TBevel;
B_Label: TSpeedButton; sb_Open: TSpeedButton;
sb_Save: TSpeedButton; sb_Preview: TSpeedButton;
sb_Book: TSpeedButton; sb_Album: TSpeedButton;
Bevel3: TBevel; sb_OLE: TSpeedButton;
sb_SnapToGrid: TSpeedButton; b_Bevel: TSpeedButton;
sb_Print: TSpeedButton; TabSheet1: TTabSheet;
TabSheet2: TTabSheet; Memo1: TMemo;
P_Font: TPanel; ColorDialog1: TColorDialog;
N3: TMenuItem; N_DeleteObject: TMenuItem;
P_HRuler: TPanel; P_Main: TPanel;
ScrollBox_: TScrollBox; ShapeSize: TShape;
P_VRuler: TPanel; TabSheet3: TTabSheet;
ImageList1: TImageList; HRuler: tmyRuler;
VRuler: tmyRuler; glBevel4: tmyBevel;
sb_FixAllMoving: TSpeedButton; sb_FixMoving: TSpeedButton;
glLabel3: TLabel; N_OLESize: TMenuItem;
N_Clip: TMenuItem; N_Center: TMenuItem;
N_Scale: TMenuItem; N_Stretch: TMenuItem;
N_AutoSize: TMenuItem; P_SBar: TPanel;
Panel5: TPanel; glLabel4: TLabel;
```

```
glLabel5: TLabel; glLabel7: TLabel;
se_Width: TSpinEdit; se_Top: TSpinEdit;
se_Left: TSpinEdit; glLabel6: TLabel;
se_Height: TSpinEdit; cb_Components: TComboBox;
glLabel8: TLabel; N4: TMenuItem;
Bevel5: TBevel; SB_Left: TSpeedButton;
SB_Bottom: TSpeedButton; SB_Right: TSpeedButton;
SB_Top: TSpeedButton; sb_AlignL: TSpeedButton;
sb_AlignC: TSpeedButton; sb_AlignR: TSpeedButton;
sb_AlignW: TSpeedButton; RxSpeedButton8: TSpeedButton;
RxSpeedButton9: TSpeedButton; sb_BevelBold: TSpeedButton;
glBevel2: tmyBevel; Panel3: TPanel;
RxSpinEdit1: TSpinEdit; Panel6: TPanel;
sbFontColor: TSpeedButton; glBevel3: tmyBevel;
Panel7: TPanel; Edit1: TMemo;
FE_OLE: TEdit; SpeedButton1: TSpeedButton;
OpenOLEFile: TOpenDialog; sb_FontUnderline: TSpeedButton;
sb_FontItalic: TSpeedButton; sb_FontBold: TSpeedButton;
TabSheet4: TTabSheet; glLabel1: TLabel;
lb_Params: tmyListBox; Panel4: TPanel;
SpeedButton2: TSpeedButton; sbBackColor: TSpeedButton;
SpeedButton3: TSpeedButton; CheckBox1: TCheckBox;
FontComboBox1: TComboBox; ColorComboBox1: TComboBox;
procedure OpenClick(Sender: TObject);
procedure Save1Click(Sender: TObject);
procedure ScrollBox_MouseDown(Sender: TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
procedure FormCreate(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure SB_LeftClick(Sender: TObject);
procedure FontComboBox1Change(Sender: TObject);
procedure RxSpinEdit1Change(Sender: TObject);
procedure ColorComboBox1Change(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure sb_BookClick(Sender: TObject);
procedure N1Click(Sender: TObject);
procedure sb_FontBoldClick(Sender: TObject);
procedure sb_AlignLClick(Sender: TObject);
procedure Memo1Change(Sender: TObject);
procedure sbFontColorClick(Sender: TObject);
procedure N_DeleteObjectClick(Sender: TObject);
procedure ScrollBox_MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure ScrollBox_MouseUp(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure FormDestroy(Sender: TObject);
procedure FormKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
procedure sb_FixMovingClick(Sender: TObject);
procedure N_AutoSizeClick(Sender: TObject);
procedure sb_SnapToGridClick(Sender: TObject);
procedure se_SizeChange(Sender: TObject);
procedure cb_ComponentsChange(Sender: TObject);
procedure N4Click(Sender: TObject);
procedure sb_BevelBoldClick(Sender: TObject);
procedure se_TopChange(Sender: TObject);
procedure se_WidthChange(Sender: TObject);
procedure se_HeightChange(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure FE_OLEChange(Sender: TObject);
procedure SpeedButton1Click(Sender: TObject);
procedure sb_PrintClick(Sender: TObject);
procedure sb_PreviewClick(Sender: TObject);
procedure SpeedButton2Click(Sender: TObject);
procedure cb_ComponentsKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
procedure CheckBox1Click(Sender: TObject);
private
ScrollBox: TlaScrollBox;
```

```
SelectedControl: tmyKHARYTONENKOReportItem;
fSelection: boolean;
SelectionRect: TRect;
procedure RemakeComponentsList;
procedure read( FileName: string; ParentWnd: TWinControl );
procedure Save( FileName: string );
procedure OnMouseDown_(Sender: TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
procedure OnMouseUp_(Sender: TObject; Button: TMouseButton; Shift: TShiftState;
X, Y: Integer);
procedure OnMouseMove_(Sender: TObject; Shift: TShiftState; X, Y: Integer);
procedure OnResize_(Sender: TObject);
procedure ShowComponentPos(Control: TControl);
procedure AssignEventsToAllComponents;
procedure UpdateToolBar( Control: tmyKHARYTONENKOReportItem);
public
Component: tmyKHARYTONENKOReport;
procedure Preview(glKHARYTONENKOReport: tmyKHARYTONENKOReport);
procedure Edit(glKHARYTONENKOReport: tmyKHARYTONENKOReport);
end;

TPublicControl = class(TControl)
public
property Caption;
end;

TPublicControlClass = class of TPublicControl;

const
IGNORE_VALUE = 65536;
var
glRepEditor: tmyRepEditor;
Form2: TComponent;

implementation
uses glTypes, glUtils;
{$R *.DFM}

procedure tmyRepEditor.OnMouseMove_(Sender: TObject; Shift: TShiftState; X, Y:
Integer);
var
DC: HDC;
i: integer;
begin
if fSelection then ScrollBox_MouseMove(Sender, Shift, X, Y);
if sb_FixAllMoving.Down then exit;
if TControl(Sender).Tag = 0 then exit;
if not fMouseDown then exit;
with ScrollBox do
for i:=0 to ControlCount-1 do
if (Controls[i] is tmyKHARYTONENKOReportItem) then with
tmyKHARYTONENKOReportItem(Controls[i]) do
if Selected and not bool(Fixed) then
begin
Left:= ((Left + X - ControlPos.x)div Step.X)*Step.X;
Top:= ((Top + Y - ControlPos.y)div Step.Y)*Step.Y;
end;
fSkipSizeUpdate:= true;
ShowComponentPos(SelectedControl);
TControl(Sender).Left:= TControl(Sender).Left + X - ControlPos.x;
TControl(Sender).Top:= TControl(Sender).Top + Y - ControlPos.y;
{
TControl(Sender).Tag:= 2;//...on moving
DC:= GetDC( TControl(Sender).Parent.Handle );
DrawFocusRect( DC, FocusRect );
FocusRect:= Bounds( TControl(Sender).Left+X-ControlPos.x,
TControl(Sender).Top+Y-ControlPos.y,  SelectedControl.Width,
SelectedControl.Height );
DrawFocusRect( DC, FocusRect );
```

```
ReleaseDC( TControl(Sender).Parent.Handle, DC );
}
end;

procedure tmyRepEditor.OnResize_(Sender: TObject);
begin
fSkipSizeUpdate:= true;
if Sender = SelectedControl then ShowComponentPos(TControl(Sender));
end;

procedure tmyRepEditor.read( FileName: string; ParentWnd: TWinControl );
begin
ScrollBox.HorzScrollBar.Position:= 0;
ScrollBox.VertScrollBar.Position:= 0;
SelectedControl:= nil;
UpdateToolBar( nil );
Component.LoadFromFile( FileName );
Component.CreateKHARYTONENKOReport( ParentWnd, true );
AssignEventsToAllComponents;
RemakeComponentsList;
end;

procedure tmyRepEditor.Save( FileName: string );
begin
ScrollBox.HorzScrollBar.Position:= 0;
ScrollBox.VertScrollBar.Position:= 0;
Component.SaveToFile( FileName );
end;

procedure tmyRepEditor.OpenClick(Sender: TObject);
begin
OpenDialog1.InitialDir:= ExtractFilePath(ParamStr(0));
if OpenDialog1.Execute then
Read( OpenDialog1.FileName, ScrollBox );
end;

procedure tmyRepEditor.Save1Click(Sender: TObject);
begin
SaveDialog1.InitialDir:= ExtractFilePath(ParamStr(0));
if SaveDialog1.Execute then Save( SaveDialog1.FileName );
end;

procedure tmyRepEditor.ScrollBox_MouseDown(Sender: TObject; Button:
TMouseButton;
Shift: TShiftState; X, Y: Integer);
var
l, Compon: tmyKHARYTONENKOReportItem;
R: TRect;
pt: TPoint;
begin
if ssCtrl in Shift then
begin
SelectionRect:=  Rect( 0,0,0,0 );
SelPt.X:= X - ScrollBox.HorzScrollBar.Position;
SelPt.Y:= Y - ScrollBox.VertScrollBar.Position;
SelPt:= ScrollBox.ClientToScreen(SelPt);
fSelection:= true;
end;
if (B_Label.Down)or(B_Bevel.Down)or(sb_OLE.Down) then
begin
Compon:= Component.AddComponent;
with Compon do
begin
Left:= X - ScrollBox.HorzScrollBar.Position;
Top:= Y - ScrollBox.VertScrollBar.Position;
if B_Label.Down then
begin
SideLeft:= 0;
SideTop := 0;
```

```
SideRight:= 0;
SideBottom:= 0;
end;
OnMouseDown:= OnMouseDown_;
OnMouseUp:= OnMouseUp_;
OnMouseMove:= OnMouseMove_;
OnResize:= OnResize_;
ContainOLE:= sb_OLE.Down;
B_Label.Down:= false;
B_Bevel.Down:= false;
sb_OLE.Down:= false;
RemakeComponentsList;
end;
end else
begin
R:= ScrollBox.ClientRect;
pt.x:= 0; pt.y:= 0; pt:= ScrollBox.ClientToScreen(pt);
OffsetRect( R, pt.x, pt.y );
ClipCursor( @R );
end;
end;

procedure tmyRepEditor.ScrollBox_MouseMove(Sender: TObject; Shift: TShiftState;
X, Y: Integer);
var
DC: HDC;
pt: TPoint;
begin

if not fSelection then exit;
DC:= GetDC( 0 );
DrawFocusRect( DC, SelectionRect );
Pt.X:= X - ScrollBox.HorzScrollBar.Position;
Pt.Y:= Y - ScrollBox.VertScrollBar.Position;
pt:= ScrollBox.ClientToScreen(pt);
SelectionRect:= Bounds( min( SelPt.X, pt.X ), min( SelPt.Y, pt.Y ), abs(
SelPt.X-pt.X ), abs( SelPt.Y-pt.Y ) );
DrawFocusRect( DC, SelectionRect );
ReleaseDC( 0, DC );
end;

procedure tmyRepEditor.sb_BookClick(Sender: TObject);
begin
if TControl(Sender).Tag=1 then
f_PrintKHARYTONENKOReport.CKHARYTONENKOReport1.Orientation:=f_PrintKHARYTONENKOR
eport.PrintWin1.Orientation else
f_PrintKHARYTONENKOReport.CKHARYTONENKOReport1.Orientation:=
f_PrintKHARYTONENKOReport.PrintWin2.Orientation;
if TControl(Sender).Tag=1 then Printer.Orientation:=poPortrait
      else Printer.Orientation:= poLandscape;
UpdatePageSize;
end;

procedure tmyRepEditor.N1Click(Sender: TObject);
begin
ScrollBox.RemoveControl( SelectedControl );
ScrollBox.InsertControl( SelectedControl );
end;

procedure tmyRepEditor.sb_FontBoldClick(Sender: TObject);
begin
if not Assigned(SelectedControl) then exit;
with SelectedControl do
case TControl(Sender).Tag of
1: FStyle:= FStyle xor 1;
2: FStyle:= FStyle xor 2;
3: FStyle:= FStyle xor 4;
end;
end;
```

```
procedure tmyRepEditor.sb_AlignLClick(Sender: TObject);
begin
if not Assigned(SelectedControl) then exit;
SelectedControl.Alignment:= TControl(Sender).Tag;
end;

procedure tmyRepEditor.sbFontColorClick(Sender: TObject);
var i: integer;
begin
if not Assigned(SelectedControl) then exit;
with ColorDialog1 do
begin
case TControl(Sender).Tag of
0: Color:= SelectedControl.FColor;
1: Color:= SelectedControl.BkColor;
else Color:= SelectedControl.BvColor;
end;

if Execute then
for i:=0 to ScrollBox.ControlCount-1 do
if ScrollBox.Controls[i] is tmyKHARYTONENKOReportItem then with
tmyKHARYTONENKOReportItem(ScrollBox.Controls[i]) do
if tmyKHARYTONENKOReportItem(ScrollBox.Controls[i]).Selected then
case TControl(Sender).Tag of
0: tmyKHARYTONENKOReportItem(ScrollBox.Controls[i]).FColor:= Color;
1: tmyKHARYTONENKOReportItem(ScrollBox.Controls[i]).BkColor:= Color;
else tmyKHARYTONENKOReportItem(ScrollBox.Controls[i]).BvColor:= Color;
end;
end;
end;

procedure tmyRepEditor.N_DeleteObjectClick(Sender: TObject);
begin
if Assigned(SelectedControl) then
begin
if Windows.MessageBox(0, 'Delete object?', 'Confirm', MB_OKCANCEL ) <> IDOK then
exit;

if SelectedControl.ContainOLE then
ScrollBox.RemoveControl( SelectedControl.OLEContainer );
ScrollBox.RemoveControl( SelectedControl );
SelectedControl.Free;
SelectedControl:= nil;
RemakeComponentsList;
end;
end;

procedure tmyRepEditor.OnDrawScrollBox(Sender: TObject);
begin
VRuler.Top:= ShapeSize.Top;
HRuler.Left:= ShapeSize.Left + P_VRuler.Width;
end;

procedure tmyRepEditor.RemakeComponentsList;
var i: integer;
begin
cb_Components.Items.Clear;
for i:= 0 to ScrollBox.ControlCount-1 do
if ScrollBox.Controls[i] is tmyKHARYTONENKOReportItem then
cb_Components.Items.Add(
tmyKHARYTONENKOReportItem(ScrollBox.Controls[i]).CompName );
cb_Components.Text:= '';
lb_Params.Items.Clear;
for i:=0 to Component.ParamNames.Count-1 do
lb_Params.Items.Add( Component.ParamNames[i] );
end;

procedure tmyRepEditor.UpdatePageSize;
```

```
const
Sizes:array[boolean,1..2] of integer = ((21,29),(29,21));
begin
ShapeSize.Width:=round(Sizes[Printer.Orientation=
poLandscape][1]*GetDeviceCaps(Canvas.Handle,LOGPIXELSX)* 1.541*2.54/10);
ShapeSize.Height:=round( Sizes[Printer.Orientation=poLandscape][2]
*GetDeviceCaps(Canvas.Handle,LOGPIXELSY)*1.541*2.54/10);
HRuler.Width:=ShapeSize.Width+10;
VRuler.Height:=ShapeSize.Height+10;
end;

procedure tmyRepEditor.FormKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
var
l, t, w, h: integer;
begin
w:= 0; h:= 0;
if (Shift = [ssCtrl])and(chr(Key)='Z')and fCanUndo then
begin
fCanUndo:=false;
ResizeKHARYTONENKOReportControls(UndoPosShift.X, UndoPosShift.Y,0,0,true);
end;

if Assigned(ActiveControl) then exit;
l:=0; t:=0;
case Key of
VK_UP: if Shift = [ssShift] then h:= -1 else if Shift = [ssCtrl] then t:= -1;
VK_DOWN: if Shift = [ssShift] then h:= 1 else if Shift = [ssCtrl] then t:= 1;
VK_LEFT: if Shift = [ssShift] then w:= -1 else if Shift = [ssCtrl] then l:= -1;
VK_RIGHT: if Shift = [ssShift] then w:= 1 else if Shift = [ssCtrl] then l:= 1;
else exit;
end;
ResizeKHARYTONENKOReportControls( l, t, w, h, true );

end;

procedure tmyRepEditor.sb_FixMovingClick(Sender: TObject);
var i: integer;
begin
with ScrollBox do
for i:=0 to ControlCount-1 do
if (Controls[i] is tmyKHARYTONENKOReportItem)and
tmyKHARYTONENKOReportItem(Controls[i]).Selected then
tmyKHARYTONENKOReportItem(Controls[i]).Fixed:= integer(sb_FixMoving.Down);
end;

procedure tmyRepEditor.N_AutoSizeClick(Sender: TObject);
begin
SelectedControl.OLESizeMode:= TMenuItem(Sender).Tag;
end;

procedure tmyRepEditor.sb_SnapToGridClick(Sender: TObject);
begin
if sb_SnapToGrid.Down then
begin Step.X:= Grid.X; Step.Y:= Grid.Y; end else
begin Step.X:= 1; Step.Y:= 1; end;
end;

procedure tmyRepEditor.se_SizeChange(Sender: TObject);
begin
if (not fMouseDown)and not fSkipSizeUpdate then
ResizeKHARYTONENKOReportControls( se_Left.Value, IGNORE_VALUE,
  IGNORE_VALUE, IGNORE_VALUE,
  false{fUseParamsAsShifts} );
ShowComponentPos(SelectedControl);
fSkipSizeUpdate:= false;
end;

procedure tmyRepEditor.se_TopChange(Sender: TObject);
```

```
begin
if (not fMouseDown)and not fSkipSizeUpdate then
ResizeKHARYTONENKOReportControls( IGNORE_VALUE, se_Top.Value,
  IGNORE_VALUE, IGNORE_VALUE,
  false{fUseParamsAsShifts} );
ShowComponentPos(SelectedControl);
fSkipSizeUpdate:= false;
end;

procedure tmyRepEditor.se_WidthChange(Sender: TObject);
begin
if (not fMouseDown)and not fSkipSizeUpdate then
ResizeKHARYTONENKOReportControls( IGNORE_VALUE, IGNORE_VALUE,
  se_Width.Value, IGNORE_VALUE,
  false{fUseParamsAsShifts} );
ShowComponentPos(SelectedControl);
fSkipSizeUpdate:= false;
end;

procedure tmyRepEditor.se_HeightChange(Sender: TObject);
begin
if (not fMouseDown)and not fSkipSizeUpdate then
ResizeKHARYTONENKOReportControls( IGNORE_VALUE, IGNORE_VALUE,
  IGNORE_VALUE, se_Height.Value,
  false{fUseParamsAsShifts} );
ShowComponentPos(SelectedControl);
fSkipSizeUpdate:= false;
end;

procedure tmyRepEditor.ResizeKHARYTONENKOReportControls( l, t, w, h: integer;
fUseParamsAsShifts: boolean );
var i: integer;
begin
with ScrollBox do
for i:=0 to ControlCount-1 do
if (Controls[i] is tmyKHARYTONENKOReportItem) then with
tmyKHARYTONENKOReportItem(Controls[i]) do
if Selected and not bool(Fixed) then
if fUseParamsAsShifts then
begin
if l < IGNORE_VALUE then Left:= Left + l; if t < IGNORE_VALUE then Top:= Top +
t;
if w < IGNORE_VALUE then Width:= Width + w; if h < IGNORE_VALUE then Height:=
Height + h;
end else
begin
if l < IGNORE_VALUE then Left:= l; if t < IGNORE_VALUE then Top:= t;
if w < IGNORE_VALUE then Width:= w; if h < IGNORE_VALUE then Height:= h;
end;
end;

procedure tmyRepEditor.cb_ComponentsChange(Sender: TObject);
var i: integer;
begin
if Assigned(SelectedControl) then
if SelectedControl.CompName = cb_Components.Text then exit;
with ScrollBox do
for i:=0 to ControlCount-1 do
if (Controls[i] is tmyKHARYTONENKOReportItem) then
if tmyKHARYTONENKOReportItem(Controls[i]).CompName = cb_Components.Text then
begin
OnMouseDown_( Controls[i], mbLeft, [], 0, 0 );
OnMouseUp_( Controls[i], mbLeft, [], 0, 0 );
exit;
end;
end;

procedure tmyRepEditor.ShowComponentPos(Control: TControl);
begin
```

```
if Component = nil then exit;
se_Left.Value:= Control.Left;
se_Top.Value:= Control.Top;
se_Width.Value:= Control.Width;
se_Height.Value:= Control.Height;
end;

procedure tmyRepEditor.AssignEventsToAllComponents;
var i: integer;
begin
with Component.ParentWnd do
for i:=0 to ControlCount-1 do
if (Controls[i] is tmyKHARYTONENKOReportItem) then with
tmyKHARYTONENKOReportItem(Controls[i]) do
begin
OnMouseDown:= OnMouseDown_;
OnMouseUp:= OnMouseUp_;
OnMouseMove:= OnMouseMove_;
OnResize:= OnResize_;
end;
end;

function CanAlignControl(Control: TControl): boolean;
begin
Result:= (Control is
tmyKHARYTONENKOReportItem)and(bool(tmyKHARYTONENKOReportItem(Control).Selected))
and(not bool(tmyKHARYTONENKOReportItem(Control).Fixed));
end;

procedure tmyRepEditor.N4Click(Sender: TObject);
begin
if not Assigned(AlignForm) then AlignForm:= TAlignForm.Create(nil);
if AlignForm.ShowModal = mrOK then
AlignControlsInWindow( Component.ParentWnd, CanAlignControl, AlignForm.Horz,
AlignForm.Vert );
end;

procedure tmyRepEditor.sb_BevelBoldClick(Sender: TObject);
var i: integer;
begin
with Component.ParentWnd do
for i:=0 to ControlCount-1 do
if (Controls[i] is tmyKHARYTONENKOReportItem)and
bool(tmyKHARYTONENKOReportItem(Controls[i]).Selected) then
tmyKHARYTONENKOReportItem(Controls[i]).PenWidth:= 1+integer(sb_BevelBold.Down);

end;

procedure tmyRepEditor.UpdateToolBar( Control: tmyKHARYTONENKOReportItem);
begin
with Control do
begin
{
se_Left.Enabled:= Assigned(Control);
se_Top.Enabled:= Assigned(Control);
se_Width.Enabled:= Assigned(Control);
se_Height.Enabled:= Assigned(Control);}
P_Sides.Enabled:= Assigned(Control);
P_Font.Enabled:= Assigned(Control);
P_SBar.Enabled:= Assigned(Control);
if not Assigned(Control) then exit;
Edit1.Text:= Text;
Memo1.Text:= Text;
FontComboBox1.Text:= FName;
RxSpinEdit1.Value:= FSize;
ColorComboBox1.Color:= FColor;

SB_Left.Down := SideLeft = 1;
SB_Top.Down  := SideTop = 1;
```

```
SB_Right.Down:= SideRight = 1;
SB_Bottom.Down:= SideBottom = 1;
case Alignment of
1: sb_AlignL.Down:= true;
2: sb_AlignR.Down:= true;
3: sb_AlignC.Down:= true;
4: sb_AlignW.Down:= true;
end;
sb_FixMoving.Down:= bool(Fixed);
sb_FontBold.Down:= boolean(FStyle and 1);
sb_FontItalic.Down:= boolean(FStyle and 2);
sb_FontUnderline.Down:= boolean(FStyle and 4);
FE_OLE.Text:= OLELinkToFile;
FE_OLE.Enabled:= ContainOLE;
sb_BevelBold.Down:= bool(PenWidth-1);
fSkipSizeUpdate:= true;
ShowComponentPos( Control );

cb_Components.Text:= CompName;

end;
end;

procedure tmyRepEditor.FormClose(Sender: TObject; var Action: TCloseAction);
var  msS, msT: TMemoryStream;
begin
if Assigned(AlignForm) then AlignForm.Free;
if Assigned(KHARYTONENKOReportParamEditor) then
KHARYTONENKOReportParamEditor.Free;
ScrollBox.HorzScrollBar.Position:= 0;
ScrollBox.VertScrollBar.Position:= 0;
Component.Save;
end;

procedure tmyRepEditor.FE_OLEChange(Sender: TObject);
var
str: string;
begin
if (not Assigned(SelectedControl))or (not FileExists(FE_OLE.Text)) then exit;
str:= FE_OLE.Text;
if ExtractFilePath(Name) = ExtractFilePath(ParamStr(0)) then
str:= ExtractFileName(Name);
if SelectedControl.OLELinkToFile <> str then
SelectedControl.OLELinkToFile:= str;
end;

procedure tmyRepEditor.SpeedButton1Click(Sender: TObject);
begin
if OpenOLEFile.Execute then FE_OLE.Text:= OpenOLEFile.FileName;
end;

procedure tmyRepEditor.sb_PrintClick(Sender: TObject);
begin
if Assigned(Component)and (Component.ComponentList.Count > 0) then
Component.Print;
Component.OwnerWnd:= self;
Component.ParentWnd:= ScrollBox;
end;

procedure tmyRepEditor.sb_PreviewClick(Sender: TObject);
var
Form: TForm;
Image: TImage;
bmp: TBitmap;
R: TRect;
i, W, H: integer;
begin
if not Assigned(Component) then exit;
Form:= TForm.Create(nil);
```

```
Form.Caption:= 'Page Preview';
Image:= TImage.Create(Form);
bmp:= TBitmap.Create;
Image.Parent:= Form;
H:= SantimsToPixels( Form.Canvas.Handle, 29, true );
W:= SantimsToPixels( Form.Canvas.Handle, 21, false );
//  Image.Width:= W+8;
Image.Left:= 0;  Image.Top:= 0;
bmp.Width:= W+7;
bmp.Height:= H+7;
try


with Component do
for i:=0 to ComponentList.Count-1 do with
tmyKHARYTONENKOReportItem(ComponentList[i]) do
begin
PaintTo(bmp.Canvas);
if ContainOle then OLEContainer.PaintTo( bmp.Canvas.Handle, Left, Top );
end;

bmp.Canvas.Brush.Color:= clBtnFace;
R:= Bounds(bmp.Width-7, 0, 7, bmp.Height-7);
bmp.Canvas.FillRect( R );
R:= Bounds(0, bmp.Height-7, bmp.Width-7, 7);
bmp.Canvas.FillRect( R );
bmp.Canvas.Brush.Color:= 0;
R:= Bounds(bmp.Width-7, 7, 7, bmp.Height-7);
bmp.Canvas.FillRect( R );
R:= Bounds(7, bmp.Height-7, bmp.Width-7, 7);
bmp.Canvas.FillRect( R );

Image.Picture.bitmap:= bmp;
Image.Stretch:= true;
Image.Width:= W div 2;
Image.Height:= H div 2;
Form.ClientWidth:= Image.Width;
Form.ClientHeight:= Image.Height;

bmp.Free; bmp:= nil;
Form.ShowModal;
finally
if Assigned(bmp) then bmp.Free;
Form.Free;
end;
end;

procedure tmyRepEditor.SpeedButton2Click(Sender: TObject);
begin
if not Assigned(KHARYTONENKOReportParamEditor) then
KHARYTONENKOReportParamEditor:= TKHARYTONENKOReportParamEditor.Create(nil);
KHARYTONENKOReportParamEditor.ShowModal;
end;

procedure tmyRepEditor.cb_ComponentsKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
begin
if (Key=VK_RETURN) then
begin
SelectedControl.CompName:= trim(cb_Components.Text);
RemakeComponentsList;
cb_Components.Text:= SelectedControl.CompName;
end;
if (Key=VK_ESCAPE) then cb_Components.Text:= SelectedControl.CompName;

end;

procedure tmyRepEditor.CheckBox1Click(Sender: TObject);
begin
```

```
if Assigned(SelectedControl) then SelectedControl.Transparent:=
integer(TCheckBox(Sender).Checked);
end;

end.
```

Файл форми виведення графічного представлення
(fset.pas)

```
unit SET;

interface

uses Windows, Messages, Classes, Forms, dialogs;

///Розробив Харитоненко Олександр Олександрович, КМ-20, ЦНТУ, 2024

type

TEMUL_LINUXCONTEINERVIRTUALIZATION = class ( TComponent )
private
FResult: boolean;
FFileName: string;
FOnTermainated: TNotifyEvent;
si: TStartupInfo;
public
pi: TProcessInformation;
function Run: boolean;
function Kill: boolean;
destructor Destroy; override;
published
property FileName: string read FFileName write FFileName;
property Result: boolean read FResult stored false;
property OnTermainated: TNotifyEvent read FOnTermainated write FOnTermainated;
end;

procedure Register;

implementation

procedure Register;
begin
RegisterComponents('Proba', [TEMUL_LINUXCONTEINERVIRTUALIZATION]);
end;


destructor TEMUL_LINUXCONTEINERVIRTUALIZATION.Destroy;
begin
Kill;
inherited;
end;

function TEMUL_LINUXCONTEINERVIRTUALIZATION.Run: boolean;
begin
GetStartupInfo(si);
si.wShowWindow:= SW_NORMAL;
FResult:= CreateProcess( PChar(FFileName), nil, nil, nil, false,
NORMAL_PRIORITY_CLASS, nil, nil, si, pi);
Run:= FResult;
if Result then
begin
while WaitForSinObjectLinuxConteinerVirtualization(pi.hProcess, 100) =
WAIT_TIMEOUT do Application.ProcessMessages;
if Assigned(OnTermainated) then OnTermainated(self);
end;
end;

function TEMUL_LINUXCONTEINERVIRTUALIZATION.Kill: boolean;
begin
if FResult {and(WaitForSingleObject(pi.hProcess, 100) <> WAIT_TIMEOUT)}
then Kill:= TerminateProcess( pi.hProcess, 0 ) else Kill:= false;
end;

end.
```

Файл форми даних розробленої програми
(data.pas)

```
unit DATA;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
dsgnintf,
  StdCtrls, glPropCn, ComCtrls, ExtCtrls, TypInfo, Buttons;

type

  Ttdata_LinuxConteinerVirtualization1 = class( TPropertyEditor )
    function GetAttributes: TPropertyAttributes; override;
    function GetValue: string; override;
    procedure Edit; override;
  end;

  Ttdata_LinuxConteinerVirtualization2 = class(TComponentEditor)
    procedure ExecuteVerb(Index: Integer); override;
    function GetVerb(Index: Integer): string; override;
    function GetVerbCount: Integer; override;
  end;


  Ttdata_LinuxConteinerVirtualization3 = class(TForm)
    lvAll: TListView;
    Label1: TLabel;
    Bevel1: TBevel;
    Bevel2: TBevel;
    Label2: TLabel;
    Bevel3: TBevel;
    Bevel4: TBevel;
    lvSel: TListView;
    pbAdd: TBitBtn;
    pbRemove: TBitBtn;
    BitBtn3: TBitBtn;
    BitBtn4: TBitBtn;
    ImageList1: TImageList;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure lvAllDblClick(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure lvSelDblClick(Sender: TObject);
    procedure lvSelChange(Sender: TObject; Item: TListItem;
      Change: TItemChange);
    procedure lvAllChange(Sender: TObject; Item: TListItem;
      Change: TItemChange);
  private
    ComponentList: TStringList;
  end;

type
  TDesigner = IDesigner;
  TFormDesigner = IFormDesigner;

var
  dataCompListEditor: Ttdata_LinuxConteinerVirtualization3;

implementation
{$R *.DFM}

procedure ShowCompListEditor(Designer: TDesigner; dataPropertyCenter:
TdataPropertyCenter);
var
```

```
  Dialog:Ttdata_LinuxConteinerVirtualization3;
  I:Integer;
begin
  Dialog:=Ttdata_LinuxConteinerVirtualization3.Create( Application );
  Dialog.Component:=dataPropertyCenter;
  Dialog.ShowModal;
  Dialog.free;
end;

function Ttdata_LinuxConteinerVirtualization1.GetAttributes:
TPropertyAttributes;
begin
  Result:=[paDialog];
end;

function Ttdata_LinuxConteinerVirtualization1.GetValue: string;
begin
  Result:= Format( '(%s)', [ GetPropType^.Name ] );
end;

procedure Ttdata_LinuxConteinerVirtualization1.Edit;
begin
  ShowCompListEditor(Designer, dataPropertyCenter(GetComponent(0)));
  GetComponent(0).Owner.Name
end;

procedure Ttdata_LinuxConteinerVirtualization2.ExecuteVerb(Index: Integer);
begin
  case Index of
    0: ShowCompListEditor(Designer, dataPropertyCenter(Component));
  end;
end;

function Ttdata_LinuxConteinerVirtualization2.GetVerbCount: Integer;
begin
  Result:= 1;
end;

procedure Ttdata_LinuxConteinerVirtualization3.FormCreate(Sender: TObject);
begin
  ControlsList:= TList.create;
end;

procedure Ttdata_LinuxConteinerVirtualization3.FormDestroy(Sender: TObject);
begin
  ControlsList.Free;
end;

procedure Ttdata_LinuxConteinerVirtualization3.FormShow(Sender: TObject);
var
  i, j: integer;
  ListItem: TListItem;
  Comp: TComponent;
  ColorPropInfo, FontPropInfo: PPropInfo;
const
  aSigns: array [boolean] of string = ('-', '+');
begin
  lvAll.Items.Clear;
  lvSel.Items.Clear;

  for i:= 0 to Component.ComponentList.Count-1 do
  begin
    ListItem:= lvSel.Items.Add;
    ListItem.Caption:= Component.ComponentList[i];
    Comp:= Component.Owner.FindComponent( Component.ComponentList[i] );
    if Comp = nil then continue;
    ColorPropInfo:= GetPropInfo( Comp.ClassInfo, 'Color');
    FontPropInfo:= GetPropInfo( Comp.ClassInfo, 'Font');
    ListItem.SubItems.Add(aSigns[Assigned(ColorPropInfo)]);
```

```
    ListItem.SubItems.Add(aSigns[Assigned(FontPropInfo)]);
    ListItem.ImageIndex:= 1;
  end;

  with Component.Owner do
  for i:=0 to ComponentCount-1 do
  begin
    ColorPropInfo:= GetPropInfo( Components[i].ClassInfo, 'Color');
    FontPropInfo:= GetPropInfo( Components[i].ClassInfo, 'Font');
    if (ColorPropInfo <> nil)or(FontPropInfo <> nil) then
     begin
       ListItem:= lvAll.Items.Add;
       ListItem.Caption:= Components[i].Name;
       ListItem.SubItems.Add(aSigns[Assigned(ColorPropInfo)]);
       ListItem.SubItems.Add(aSigns[Assigned(FontPropInfo)]);

       for j:=0 to lvSel.Items.Count - 1 do
         if lvSel.Items[j].Caption = ListItem.Caption then
         begin ListItem.ImageIndex:= 1; break; end;

     end;
     SetOrdProp( FormX.Components[i], PropInfo, clGreen );
  end;
end;

procedure Ttdata_LinuxConteinerVirtualization3.lvAllDblClick(Sender: TObject);
var ListItem: TListItem;
begin
  if (lvAll.Selected = nil)or(lvAll.Selected.ImageIndex = 1) then exit;

  ListItem:= lvSel.Items.Add;
  ListItem.Caption:= lvAll.Selected.Caption;
  ListItem.SubItems.Add(lvAll.Selected.SubItems[0]);
  ListItem.SubItems.Add(lvAll.Selected.SubItems[1]);
  lvAll.Selected.ImageIndex:= 1;
  ListItem.ImageIndex:= 1;
end;

procedure Ttdata_LinuxConteinerVirtualization3.BitBtn4Click(Sender: TObject);
begin close; end;

procedure Ttdata_LinuxConteinerVirtualization3.BitBtn3Click(Sender: TObject);
var
  i:integer;
  Comp:TComponent;
begin
  Component.ComponentList.Clear;
  Component.CompList.Clear;
  for i:=0 to lvSel.Items.Count-1 do
  begin
  Comp:=Component.Owner.FindComponent(lvSel.Items[i].Caption);
    if Comp = nil then continue;
    Component.CompList.Add(Comp);
    Component.ComponentList.Add(lvSel.Items[i].Caption);
  end;
  close;
end;

procedure Ttdata_LinuxConteinerVirtualization3.lvSelDblClick(Sender: TObject);
var i: integer;
begin
  if not Assigned(lvSel.Selected) then exit;
  for i:=0 to lvAll.Items.Count - 1 do
  if lvAll.Items[i].Caption = lvSel.Selected.Caption then break;
  lvAll.Items[i].ImageIndex:= 0;
  lvSel.Items.Delete(lvSel.Selected.Index);
end;
```

```
procedure Ttdata_LinuxConteinerVirtualization3.lvSelChange(Sender:
TObject;Item:TListItem;Change:TItemChange);
begin
  pbRemove.Enabled:=Assigned(lvSel.Selected);
end;

procedure Ttdata_LinuxConteinerVirtualization3.lvAllChange(Sender: TObject;
Item: TListItem; Change: TItemChange);
begin
pbAdd.Enabled:=Assigned(lvAll.Selected)and(lvAll.Selected.ImageIndex=0);
end;

end.
```

Файл форми роботи программної моделі
(LinuxConteinerVirtualizationEmulation.pas)

```
unit LinuxConteinerVirtualizationEmulation;
///Розробив Харитоненко Олександр Олександрович, КМ-20, ЦНТУ, 2024
interface
uses Windows,Graphics,Controls,Classes,ExtCtrls,glTypes,SysUtils;

type
TTwainColors = class; Tem_LinuxConteinerVirtualization_CustomLabelColors =
class;
Tem_LinuxConteinerVirtualization_LabelColors = class; TCustomGradient = class;
TGradient = class; TThreeDGradient = class;
T2DAlign = class; Tem_LinuxConteinerVirtualization_CustomBoxStyle= class;
Tem_LinuxConteinerVirtualization_CustomTextBoxStyle= class;
Tem_LinuxConteinerVirtualization_TextBoxStyle = class;
TPointClass = class; Tem_LinuxConteinerVirtualization_Bevel = class;
Tem_LinuxConteinerVirtualization_ExtBevel = class; TIllumination = class;
Tem_LinuxConteinerVirtualization_LabelTextStyles = class;
Tem_LinuxConteinerVirtualization_CustomTextColors = class;
Tem_LinuxConteinerVirtualization_SimleLabelColors = class;
Tem_LinuxConteinerVirtualization_BevelLines = class; TTwainColors =
class(TPersistent)
Tem_LinuxConteinerVirtualization_GrBoxColors = class;
Tem_LinuxConteinerVirtualization_ListBoxItemStyle  = class;
Tem_LinuxConteinerVirtualization_AskListBoxItemStyle= class;
private
  FFromColor:TColor; FToColor:TColor;
  procedure SeTem_LinuxConteinerVirtualization_omColor( Value:TColor );
  procedure SetToColor( Value:TColor );
public
  FRGBFromColor:Longint;
  FRGBToColor:Longint;
  OnChanged:TNotifyEvent;
  constructor Create; virtual;
published
  property FromColor:TColor read FFromColor write
SeTem_LinuxConteinerVirtualization_omColor default $00808080;
   property ToColor:TColor read FToColor write SetToColor
default 0;
end;

TCustomGradient = class(TTwainColors)
private
  FBufferedDraw: boolean;
  FSteps: integer;
  FPercentFilling: Tem_LinuxConteinerVirtualization_Percent;
  FBrushStyle: TBrushStyle;
procedure SetActive( Value:boolean );
procedure SetOrientation( Value:Tem_LinuxConteinerVirtualization_GradientDir );
procedure SetSteps( Value: integer );
procedure SetPercentFilling( Value: Tem_LinuxConteinerVirtualization_Percent );
procedure SetBrushStyle( Value: TBrushStyle );
public
FOrientation: Tem_LinuxConteinerVirtualization_GradientDir;
FActive: boolean;
fReverse: boolean;
procedure TextOut(DC:HDC;Str:string; TextR:TRect; x, y:integer);
function GetColorFromGradientLine( GradientLineWidth, Position: word ):
COLORREF;
constructor Create; override;
destructor Destroy;override;
protected
property Active:boolean read FActive write SetActive;
property BufferedDraw:boolean read FBufferedDraw write FBufferedDraw
default false;
property Orientation:Tem_LinuxConteinerVirtualization_GradientDir read
FOrientation write SetOrientation;
property Steps: integer read FSteps write SetSteps default 255;
```

```
property PercentFilling: Tem_LinuxConteinerVirtualization_Percent read
FPercentFilling write SetPercentFilling
default 100;
property BrushStyle: TBrushStyle read FBrushStyle write SetBrushStyle
default bsSolid;
end;
TGradient = class(TCustomGradient)
private
public
procedure Draw(DC: HDC; r: TRect; PenStyle, PenWidth: integer);
published
property Active;
property BufferedDraw;
property Orientation;
property Steps;
property PercentFilling;
property BrushStyle;
end;
TThreeDGradient = class(TCustomGradient)
private
FDepth:word;
FGType:TThreeDGradientType;
procedure SetDepth(Value: word);
procedure SetGType(Value: TThreeDGradientType);
public
constructor Create; override;
published
property Depth: word
read FDepth write SetDepth default 16;
property GType:TThreeDGradientType read FGType write SetGType default fgtFlat;
end;
T2DAlign = class(TPersistent)
private
FHorizontal: Tem_LinuxConteinerVirtualization_HorAlign;
FVertical: Tem_LinuxConteinerVirtualization_VertAlign;
procedure SetHorizontal(Value: Tem_LinuxConteinerVirtualization_HorAlign);
procedure SetVertical(Value: Tem_LinuxConteinerVirtualization_VertAlign);
public
OnChanged:TNotifyEvent;
constructor Create;
published
property Horizontal:Tem_LinuxConteinerVirtualization_HorAlign read FHorizontal
write SetHorizontal
default fhaLeft;
property Vertical:Tem_LinuxConteinerVirtualization_VertAlign read FVertical
write SetVertical
default fvaTop;
end;
TPointClass = class(TPersistent)
private
FX: integer;
FY: integer;
procedure SetX(Value: integer);
procedure SetY(Value: integer);
public
OnChanged:TNotifyEvent;
published
property X:integer read FX write SetX;
property Y:integer read FY write SetY;
end;
Tem_LinuxConteinerVirtualization_Bevel = class(TPersistent)
private
FInner: TPanelBevel;
FOuter: TPanelBevel;
FSides: Tem_LinuxConteinerVirtualization_Sides;
FBold: boolean;
procedure SetInner(Value: TPanelBevel);
procedure SetOuter(Value: TPanelBevel);
procedure SetSides(Value: Tem_LinuxConteinerVirtualization_Sides);
```

```
procedure SetBold(Value: boolean);
public
OnChanged:TNotifyEvent;
constructor Create; virtual;
function BordersHeight: integer;
function BordersWidth: integer;
published
property Inner: TPanelBevel read FInner write SetInner stored true;//    default
bvLowered;
property Outer: TPanelBevel read FOuter write SetOuter stored true;//    default
bvNone;
property Sides: Tem_LinuxConteinerVirtualization_Sides read FSides write
SetSides stored true default ALLGLSIDES;
property Bold: boolean read FBold write SetBold stored true;//  default false;
end;
Tem_LinuxConteinerVirtualization_ExtBevel =
class(Tem_LinuxConteinerVirtualization_Bevel)
private
FActive: boolean;
FBevelPenStyle: TPenStyle;
FBevelPenWidth: word;
FInteriorOffset: word;
procedure SetActive(Value: boolean);
procedure SetBevelPenStyle(Value: TPenStyle);
procedure SetBevelPenWidth(Value: word);
procedure SetInteriorOffset(Value: word);
public
constructor Create; override;
published
property Active: boolean read FActive write SetActive
default true;
property BevelPenStyle: TPenStyle read FBevelPenStyle write SetBevelPenStyle
default psSolid;
property BevelPenWidth: word read FBevelPenWidth write SetBevelPenWidth
default 1;
property InteriorOffset: word read FInteriorOffset write SetInteriorOffset
default 0;
end;
TIllumination = class(T2DAlign)
private
procedure SetShadowDepth(Value: integer);
public
FShadowDepth: integer;
OnChanged:TNotifyEvent;
constructor Create;
published
property ShadowDepth: integer
read FShadowDepth write SetShadowDepth default 2;
end;
Tem_LinuxConteinerVirtualization_LabelTextStyles = class(TPersistent)
private
FPassive: Tem_LinuxConteinerVirtualization_TextStyle;
FActive: Tem_LinuxConteinerVirtualization_TextStyle;
FDisabled: Tem_LinuxConteinerVirtualization_TextStyle;
procedure SetPassive(Value: Tem_LinuxConteinerVirtualization_TextStyle);
procedure SetActive(Value: Tem_LinuxConteinerVirtualization_TextStyle);
procedure SetDisabled(Value: Tem_LinuxConteinerVirtualization_TextStyle);
public
OnChanged:TNotifyEvent;
constructor Create;
published
property Passive: Tem_LinuxConteinerVirtualization_TextStyle read FPassive write
SetPassive
default fstRaised;
property Active: Tem_LinuxConteinerVirtualization_TextStyle read FActive write
SetActive
default fstRaised;
property Disabled: Tem_LinuxConteinerVirtualization_TextStyle read FDisabled
write SetDisabled
```

```
default fstPushed;
end;
Tem_LinuxConteinerVirtualization_CustomTextColors = class(TPersistent)
private
FText: TColor;
FTextDisabled: TColor;
FDelineate: TColor;
FBackground: TColor;
public
FHighlight: TColor;
FShadow: TColor;
private
procedure SetText(Value: TColor);
procedure SetTextDisabled(Value: TColor);
procedure SetDelineate(Value: TColor);
procedure SetBackground(Value: TColor);
procedure SetHighlight(Value: TColor);
procedure SetShadow(Value: TColor);
public
OnChanged:TNotifyEvent;
constructor Create; virtual;
protected
property Text: TColor read FText write SetText
default clBlack;
property TextDisabled: TColor read FTextDisabled write SetTextDisabled
default clGray;
property Delineate: TColor read FDelineate write SetDelineate
default clWhite;
property Shadow:TColor read FShadow write SetShadow default clBtnShadow;
property Highlight:TColor read FHighlight write SetHighlight default
clBtnHighlight;
property Background:TColor read FBackground write SetBackground default
clBtnFace;
end;
Tem_LinuxConteinerVirtualization_SimleLabelColors =
class(Tem_LinuxConteinerVirtualization_CustomTextColors)
published
property Text stored true;
property Delineate stored true;
property Shadow stored true;
property Highlight;
property Background stored true;
end;
Tem_LinuxConteinerVirtualization_CustomLabelColors =
class(Tem_LinuxConteinerVirtualization_CustomTextColors)
private
FTextActive: TColor;
FDelineateActive: TColor;
FAutoHighlight: boolean;
FAutoShadow: boolean;
FBackgroundActive: TColor;
public
FColorHighlightShift: integer;
FColorShadowShift: integer;
private
procedure SetTextActive(Value: TColor);
procedure SetDelineateActive(Value: TColor);
procedure SetBackgroundActive(Value: TColor);
procedure SetAutoHighlight(Value: boolean);
procedure SetAutoShadow(Value: boolean);
procedure SetColorHighlightShift(Value: integer);
procedure SetColorShadowShift(Value: integer);
public
OnChanged:TNotifyEvent;
constructor Create; override;
protected
property TextActive: TColor read FTextActive write SetTextActive
default clBlack;
property DelineateActive: TColor read FDelineateActive write SetDelineateActive
```

```
default clWhite;
property AutoHighlight: boolean
read FAutoHighlight write SetAutoHighlight default false;
property AutoShadow: boolean
read FAutoShadow write SetAutoShadow default false;
property ColorHighlightShift: integer
read FColorHighlightShift write SetColorHighlightShift default 40;
property ColorShadowShift: integer
read FColorShadowShift write SetColorShadowShift default 60;
property BackgroundActive: TColor
read FBackgroundActive write SetBackgroundActive default clBtnFace;
end;
Tem_LinuxConteinerVirtualization_LabelColors =
class(Tem_LinuxConteinerVirtualization_CustomLabelColors)
published
property Text;
property TextDisabled;
property Delineate;
property Shadow;
property Highlight;
property Background;
property TextActive;
property DelineateActive;
property AutoHighlight;
property AutoShadow;
property ColorHighlightShift;
property ColorShadowShift;
property BackgroundActive;
end;
Tem_LinuxConteinerVirtualization_GrBoxColors =
class(Tem_LinuxConteinerVirtualization_CustomLabelColors)
private
FCaption: TColor;
FCaptionActive: TColor;
FClient: TColor;
FClientActive: TColor;
procedure SetCaption(Value: TColor);
procedure SetCaptionActive(Value: TColor);
procedure SetClient(Value: TColor);
procedure SetClientActive(Value: TColor);
public
constructor Create; override;
published
property Text;
property Delineate;
property Shadow;
property Highlight;
property Background;
property TextActive;
property DelineateActive;
property AutoHighlight;
property AutoShadow;
property ColorHighlightShift;
property ColorShadowShift;
property BackgroundActive;
property Caption: TColor read FCaption write SetCaption;
property CaptionActive: TColor read FCaptionActive write SetCaptionActive;
property Client: TColor read FClient write SetClient;
property ClientActive: TColor read FClientActive write SetClientActive;
end;
Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle  = class(TPersistent)
private
FColor: TColor;
FDelineateColor: TColor;
FFont: TFont;
FBevel: Tem_LinuxConteinerVirtualization_Bevel;
FTextStyle: Tem_LinuxConteinerVirtualization_TextStyle;
FOnChanged: TNotifyEvent;
procedure SetColor(Value: TColor);
```

```
procedure SetDelineateColor(Value: TColor);
procedure SetFont(Value: TFont);
procedure SetTextStyle(Value: Tem_LinuxConteinerVirtualization_TextStyle);
protected
procedure SetOnChanged(Value: TNotifyEvent); virtual;
public
property OnChanged: TNotifyEvent read FOnChanged write SetOnChanged;
constructor Create; virtual;
destructor Destroy; override;
function HighlightColor: TColor;
function ShadowColor: TColor;
published
property Color: TColor read FColor write SetColor;
property DelineateColor: TColor read FDelineateColor write SetDelineateColor;
property Font: TFont read FFont write SetFont;
property Bevel: Tem_LinuxConteinerVirtualization_Bevel read FBevel write FBevel;
property TextStyle: Tem_LinuxConteinerVirtualization_TextStyle read FTextStyle
write SetTextStyle;
end;
Tem_LinuxConteinerVirtualization_ListBoxItemStyle=class(Tem_LinuxConteinerVirtua
lization_CustomListBoxItemStyle)
private
FGradient: TGradient;
FTextGradient: TGradient;
protected
property TextGradient: TGradient read FTextGradient write FTextGradient;
procedure SetOnChanged(Value: TNotifyEvent); override;
public
constructor Create; override;
destructor Destroy; override;
published
property Gradient: TGradient read FGradient write FGradient;
end;

Tem_LinuxConteinerVirtualization_HintStyle=class(Tem_LinuxConteinerVirtualizatio
n_ListBoxItemStyle)
end;

Tem_LinuxConteinerVirtualization_SpeedButtonStyle=class(Tem_LinuxConteinerVirtua
lization_ListBoxItemStyle)
published
property TextGradient;
end;
Tem_LinuxConteinerVirtualization_AskListBoxItemStyle=
class(Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle)
private
FBtnColor: TColor;
FBtnFont: TFont;
FBtnTextStyle: Tem_LinuxConteinerVirtualization_TextStyle;
procedure SetBtnColor(Value: TColor);
procedure SetBtnFont(Value: TFont);
procedure SetBtnTextStyle(Value: Tem_LinuxConteinerVirtualization_TextStyle);
public
constructor Create; override;
destructor Destroy; override;
published
property BtnColor: TColor read FBtnColor write SetBtnColor;
property BtnFont: TFont read FBtnFont write SetBtnFont;
property BtnTextStyle: Tem_LinuxConteinerVirtualization_TextStyle read
FBtnTextStyle write SetBtnTextStyle;
end;
Tem_LinuxConteinerVirtualization_CustomBoxStyle=
class(Tem_LinuxConteinerVirtualization_Bevel)
private
FPenStyle: TPenStyle;
FHighlightColor: TColor;
FShadowColor: TColor;
procedure SetPenStyle(Value: TPenStyle);
procedure SetHighlightColor(Value: TColor);
```

```
procedure SetShadowColor(Value: TColor);
public
OnChanged: TNotifyEvent;
constructor Create; override;
protected
property PenStyle: TPenStyle read FPenStyle write SetPenStyle
default psSolid;
property HighlightColor: TColor read FHighlightColor write SetHighlightColor
default clBtnHighlight;
property ShadowColor: TColor read FShadowColor write SetShadowColor
default clBtnShadow;
end;
Tem_LinuxConteinerVirtualization_CustomTextBoxStyle=
class(Tem_LinuxConteinerVirtualization_CustomBoxStyle)
private
FTextColor: TColor;
FBackgroundColor: TColor;
procedure SetTextColor(Value: TColor);
procedure SetBackgroundColor(Value: TColor);
public
constructor Create; override;
protected
property TextColor: TColor read FTextColor write SetTextColor
default clBlack;
property BackgroundColor: TColor read FBackgroundColor write SetBackgroundColor
default clWindow;
end;
Tem_LinuxConteinerVirtualization_TextBoxStyle =
class(Tem_LinuxConteinerVirtualization_CustomTextBoxStyle)
published
property Inner;
property Outer;
property Sides;
property Bold;
property PenStyle;
property TextColor;
property BackgroundColor;
property HighlightColor;
property ShadowColor;
end;
Tem_LinuxConteinerVirtualization_BevelLines = class(TPersistent)
private
FCount: cardinal;
FStep: cardinal;
FOrigin: Tem_LinuxConteinerVirtualization_Origin;
FStyle: TPanelBevel;
FBold: boolean;
FThickness: byte;
FIgnoreBorder: boolean;

procedure SetCount(Value: cardinal);
procedure SetStep(Value: cardinal);
procedure SetOrigin(Value: Tem_LinuxConteinerVirtualization_Origin);
procedure SetStyle(Value: TPanelBevel);
procedure SetBold(Value: boolean);
procedure SetThickness(Value: byte);
procedure SetIgnoreBorder(Value: boolean);
public
OnChanged: TNotifyEvent;
constructor Create;
published
property Count: cardinal read FCount write SetCount
default 0;
property Step: cardinal read FStep write SetStep
default 0;
property Origin: Tem_LinuxConteinerVirtualization_Origin read FOrigin write
SetOrigin
default forLeftTop;
property Style: TPanelBevel read FStyle write SetStyle
```

```
default bvLowered;
property Bold: boolean read FBold write SetBold
default false;
property Thickness: byte read FThickness write SetThickness
default 1;
property IgnoreBorder: boolean read FIgnoreBorder write SetIgnoreBorder
default false;
end;
implementation

{$I Debug.inc}

constructor TTwainColors.Create;
begin
inherited Create;
FFromColor:= $00808080; FRGBFromColor:= ColorToRGB( FFromColor );
FToColor:= 0;FRGBToColor:= ColorToRGB( FToColor );
end;

procedure TTwainColors.SeTem_LinuxConteinerVirtualization_omColor( Value:TColor
);
begin
if FFromColor = Value then exit;
FFromColor:= Value; FRGBFromColor:= ColorToRGB( Value );
if Assigned(OnChanged) then OnChanged(self);
end;

procedure TTwainColors.SetToColor( Value:TColor );
begin
if FToColor = Value then exit;
FToColor:= Value; FRGBToColor:= ColorToRGB( Value );
if Assigned(OnChanged) then OnChanged(self);
end;
constructor TCustomGradient.Create;
begin
inherited Create;
FActive:= false;
FBufferedDraw:= false;
FOrientation:= fgdHorizontal;
FSteps:= 255;
FPercentFilling:= 100;
FBrushStyle:= bsSolid;
end;

destructor TCustomGradient.Destroy;
begin inherited Destroy; end;

procedure TCustomGradient.Free;
begin if self<>nil then Destroy; end;

procedure TCustomGradient.SetActive( Value:boolean );
begin
if FActive = Value then exit;
FActive:= Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure TCustomGradient.SetOrientation(
Value:Tem_LinuxConteinerVirtualization_GradientDir );
begin
if FOrientation = Value then exit;
FOrientation:= Value;
if FActive and Assigned(OnChanged) then OnChanged(self);
end;

procedure TCustomGradient.SetSteps( Value: integer );
begin
if Value > 255 then Value:= 255
else if Value < 1 then Value:= 1;
```

```
if FSteps = Value then exit;
FSteps:= Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure TCustomGradient.SetPercentFilling( Value:
Tem_LinuxConteinerVirtualization_Percent );
begin
if FPercentFilling = Value then exit;
FPercentFilling:= Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure TCustomGradient.SetBrushStyle( Value: TBrushStyle );
begin
FBrushStyle:= Value;
if Assigned(OnChanged) then OnChanged(self);
end;

function TCustomGradient.GetColorFromGradientLine
( GradientLineWidth, Position: word ): COLORREF;
var
c1F,c2F,c3F:byte;
c1T,c2T,c3T:byte;
Step1,Step2,Step3:SinEm_LinuxConteinerVirtualization;
begin
c1F:= Byte( self.FRGBFromColor ) ;
c2F:= Byte( WORD(self.FRGBFromColor) shr 8);
c3F:= Byte( self.FRGBFromColor shr 16 );
c1T:= Byte( self.FRGBToColor ) ;
c2T:= Byte( WORD(self.FRGBToColor) shr 8);
c3T:= Byte( self.FRGBToColor shr 16 );

Step1:=(c1T-c1F)/GradientLineWidth;
Step2:=(c2T-c2F)/GradientLineWidth;
Step3:=(c3T-c3F)/GradientLineWidth;

Result:=RGB( trunc(c1F+Step1*Position),
trunc(c2F+Step2*Position),
trunc(c3F+Step3*Position) );
end;


procedure TCustomGradient.TextOut( DC: HDC; Str:string; TextR:TRect; x,
y:integer );
var
i,Steps:integer;
r:TRect;
c1F,c2F,c3F:byte;
c1T,c2T,c3T:byte;
c1,c2,c3:SinEm_LinuxConteinerVirtualization;
OldTextColor:TCOLORREF;
begin
if (not Active)or(GetDeviceCaps( DC, BITSPIXEL )<16) then
begin Windows.TextOut( DC, x,y, PChar(str), Length(str)); exit; end;
r:= TextR;
c1F:= Byte( FRGBFromColor ) ;
c2F:= Byte( WORD(FRGBFromColor) shr 8);
c3F:= Byte( FRGBFromColor shr 16 );
c1T:= Byte( FRGBToColor ) ;
c2T:= Byte( WORD(FRGBToColor) shr 8);
c3T:= Byte( FRGBToColor shr 16 );

c1:=c1F; c2:=c2F; c3:=c3F;
if FOrientation = fgdVertical then Steps:=r.right-r.left
else Steps:=r.bottom-r.top;
Step1:=(c1T-c1F)/Steps; Step2:=(c2T-c2F)/Steps; Step3:=(c3T-c3F)/Steps;

OldTextColor:= SetTextColor( DC, 0 );
```

```
Steps:= MulDiv( Steps, PercentFilling, 100 );
for i:=0 to Steps do
begin
SetTextColor( DC, RGB( trunc(c1),trunc(c2),trunc(c3)) );

if FOrientation = fgdVertical then
begin r.left:=i; r.right:=r.left+1; end
else
begin r.top:=i; r.bottom:=r.top+1; end;

Windows.ExtTextOut( DC, x, y,  ETO_CLIPPED, @r,
PChar(str), Length(str), nil);
c1:= c1+ Step1; c2:= c2+ Step2; c3:= c3+ Step3;
end;
SetTextColor( DC, OldTextColor );
end;
constructor TThreeDGradient.Create;
begin
inherited Create;
Depth:=16;
FGType:=fgtFlat;
FActive:=true;
end;

procedure TThreeDGradient.SetGType(Value: TThreeDGradientType);
begin
if FGType = Value then exit;
FGType:= Value; if FActive and Assigned(OnChanged) then OnChanged(self);
end;

procedure TThreeDGradient.SetDepth(Value: word);
begin
if FDepth = Value then exit;
FDepth:= Value; if FActive and Assigned(OnChanged) then OnChanged(self);
end;

constructor T2DAlign.Create;
begin
inherited Create;
FHorizontal:= fhaLeft;
FVertical:= fvaTop;
end;

procedure T2DAlign.SetHorizontal(Value:
Tem_LinuxConteinerVirtualization_HorAlign);
begin
if FHorizontal = Value then exit; FHorizontal:= Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure T2DAlign.SetVertical(Value:
Tem_LinuxConteinerVirtualization_VertAlign);
begin
if FVertical = Value then exit; FVertical:= Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure TPointClass.SetX(Value: integer);
begin
if FX = Value then exit; FX:= Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure TPointClass.SetY(Value: integer);
begin
if FY = Value then exit; FY:= Value;
if Assigned(OnChanged) then OnChanged(self);
end;
```

```
constructor Tem_LinuxConteinerVirtualization_Bevel.Create;
begin
inherited;
FSides:= ALLGLSIDES;
end;

procedure Tem_LinuxConteinerVirtualization_Bevel.SetOuter(Value: TPanelBevel);
begin
if FOuter=Value then exit; FOuter:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_Bevel.SetInner(Value: TPanelBevel);
begin
if FInner=Value then exit; FInner:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_Bevel.SetSides(Value:
Tem_LinuxConteinerVirtualization_Sides);
begin
if FSides=Value then exit; FSides:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_Bevel.SetBold(Value: boolean);
begin
if FBold=Value then exit; FBold:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

constructor TIllumination.Create;
begin
inherited;
FShadowDepth:= 2;
end;

procedure TIllumination.SetShadowDepth(Value: integer);
begin
if Value < 0 then Value:=0;
if FShadowDepth=Value then exit; FShadowDepth:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

constructor Tem_LinuxConteinerVirtualization_LabelTextStyles.Create;
begin
inherited;
FActive:= fstRaised;
FPassive:= fstRaised;
FDisabled:= fstPushed;
end;

procedure Tem_LinuxConteinerVirtualization_LabelTextStyles.SetPassive(Value:
Tem_LinuxConteinerVirtualization_TextStyle);
begin
if FPassive=Value then exit; FPassive:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_LabelTextStyles.SetActive(Value:
Tem_LinuxConteinerVirtualization_TextStyle);
begin
if FActive=Value then exit; FActive:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_LabelTextStyles.SetDisabled(Value:
Tem_LinuxConteinerVirtualization_TextStyle);
begin
```

```
if FDisabled=Value then exit; FDisabled:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

constructor Tem_LinuxConteinerVirtualization_CustomTextColors.Create;
begin
inherited;
FText:= clBlack;
FTextDisabled:= clGray;
FDelineate:= clWhite;
FHighlight:= clBtnHighlight;
FShadow:= clBtnShadow;
FBackground:= clBtnFace;
end;

procedure Tem_LinuxConteinerVirtualization_CustomTextColors.SetText(Value:
TColor);
begin
if FText=Value then exit; FText:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure
Tem_LinuxConteinerVirtualization_CustomTextColors.SetTextDisabled(Value:
TColor);
begin
if FTextDisabled=Value then exit; FTextDisabled:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_CustomTextColors.SetDelineate(Value:
TColor);
begin
if FDelineate=Value then exit; FDelineate:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_CustomTextColors.SetHighlight(Value:
TColor);
begin
if FHighlight=Value then exit; FHighlight:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_CustomTextColors.SetShadow(Value:
TColor);
begin
if FShadow=Value then exit; FShadow:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_CustomTextColors.SetBackground(Value:
TColor);
begin
if FBackground=Value then exit; FBackground:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

constructor Tem_LinuxConteinerVirtualization_CustomLabelColors.Create;
begin
inherited;
FTextActive:= clBlack;
FDelineateActive:= clWhite;
FAutoHighlight:= false;
FAutoShadow:= false;
FColorHighlightShift:= 40;
FColorShadowShift:= 60;
FBackgroundActive:= clBtnFace;
end;
```

```
procedure
Tem_LinuxConteinerVirtualization_CustomLabelColors.SetTextActive(Value: TColor);
begin
if FTextActive=Value then exit; FTextActive:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure
Tem_LinuxConteinerVirtualization_CustomLabelColors.SetDelineateActive(Value:
TColor);
begin
if FDelineateActive=Value then exit; FDelineateActive:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure
Tem_LinuxConteinerVirtualization_CustomLabelColors.SetAutoHighlight(Value:
boolean);
begin
if FAutoHighlight=Value then exit; FAutoHighlight:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure
Tem_LinuxConteinerVirtualization_CustomLabelColors.SetAutoShadow(Value:
boolean);
begin
if FAutoShadow=Value then exit; FAutoShadow:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure
Tem_LinuxConteinerVirtualization_CustomLabelColors.SetColorHighlightShift(Value:
integer);
begin
if FColorHighlightShift=Value then exit; FColorHighlightShift:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure
Tem_LinuxConteinerVirtualization_CustomLabelColors.SetColorShadowShift(Value:
integer);
begin
if FColorShadowShift=Value then exit; FColorShadowShift:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

procedure
Tem_LinuxConteinerVirtualization_CustomLabelColors.SetBackgroundActive(Value:
TColor);
begin
if FBackgroundActive=Value then exit; FBackgroundActive:=Value;
if Assigned(OnChanged) then OnChanged(self);
end;

constructor Tem_LinuxConteinerVirtualization_GrBoxColors.Create;
begin
inherited;
FCaption:= clBtnFace;
FCaptionActive:= clBtnFace;
FClient:= clBtnFace;
FClientActive:= clBtnFace;
end;

procedure Tem_LinuxConteinerVirtualization_GrBoxColors.SetCaption(Value:
TColor);
begin FCaption:= Value; if Assigned(OnChanged) then OnChanged(self); end;
```

```
procedure Tem_LinuxConteinerVirtualization_GrBoxColors.SetCaptionActive(Value:
TColor);
begin FCaptionActive:= Value; if Assigned(OnChanged) then OnChanged(self); end;

procedure Tem_LinuxConteinerVirtualization_GrBoxColors.SetClient(Value: TColor);
begin FClient:= Value; if Assigned(OnChanged) then OnChanged(self); end;

procedure Tem_LinuxConteinerVirtualization_GrBoxColors.SetClientActive(Value:
TColor);
begin FClientActive:= Value; if Assigned(OnChanged) then OnChanged(self); end;

constructor Tem_LinuxConteinerVirtualization_ExtBevel.Create;
begin
inherited;
FActive:= true;
FBevelPenStyle:= psSolid;
FBevelPenWidth:= 1;
end;

procedure Tem_LinuxConteinerVirtualization_ExtBevel.SetActive(Value: boolean);
begin
if FActive = Value then exit;
FActive:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_ExtBevel.SetBevelPenStyle(Value:
TPenStyle);
begin
if FBevelPenStyle = Value then exit;
FBevelPenStyle:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_ExtBevel.SetBevelPenWidth(Value:
word);
begin
if FBevelPenWidth = Value then exit;
FBevelPenWidth:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_ExtBevel.SetInteriorOffset(Value:
word);
begin
if FInteriorOffset = Value then exit;
FInteriorOffset:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

constructor Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle.Create;
begin
inherited Create;
FBevel:= Tem_LinuxConteinerVirtualization_Bevel.Create;
FFont:= TFont.Create;
end;

destructor Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle.Destroy;
begin
FFont.Free; FBevel.Free; inherited;
end;

procedure
Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle.SetOnChanged(Value:
TNotifyEvent);
begin
FOnChanged:= Value; FBevel.OnChanged:= Value;
end;

procedure
Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle.SetColor(Value: TColor);
begin
if FColor = Value then exit;
```

```
FColor:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

procedure
Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle.SetDelineateColor(Value:
TColor);
begin
if FDelineateColor = Value then exit;
FDelineateColor:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle.SetFont(Value:
TFont);
begin
FFont.Assign(Value); if Assigned(OnChanged) then OnChanged(self);
end;

procedure
Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle.SetTextStyle(Value:
Tem_LinuxConteinerVirtualization_TextStyle);
begin
FTextStyle:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

constructor Tem_LinuxConteinerVirtualization_ListBoxItemStyle.Create;
begin
inherited Create;
FGradient:= TGradient.Create;
FTextGradient:= TGradient.Create;
end;

destructor Tem_LinuxConteinerVirtualization_ListBoxItemStyle.Destroy;
begin
FGradient.Free;
FTextGradient.Free;
inherited;
end;

procedure Tem_LinuxConteinerVirtualization_ListBoxItemStyle.SetOnChanged(Value:
TNotifyEvent);
begin
inherited SetOnChanged(Value);
FGradient.OnChanged:= Value;
end;

constructor Tem_LinuxConteinerVirtualization_AskListBoxItemStyle.Create;
begin
inherited Create;
FBtnFont:= TFont.Create;
end;

destructor Tem_LinuxConteinerVirtualization_AskListBoxItemStyle.Destroy;
begin
FBtnFont.Free; inherited;
end;

procedure
Tem_LinuxConteinerVirtualization_AskListBoxItemStyle.SetBtnColor(Value: TColor);
begin
if FBtnColor = Value then exit;
FBtnColor:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_AskListBoxItemStyle.SetBtnFont(Value:
TFont);
begin
FBtnFont.Assign(Value); if Assigned(OnChanged) then OnChanged(self);
end;
```

```
procedure
Tem_LinuxConteinerVirtualization_AskListBoxItemStyle.SetBtnTextStyle(Value:
Tem_LinuxConteinerVirtualization_TextStyle);
begin
FBtnTextStyle:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

constructor Tem_LinuxConteinerVirtualization_CustomBoxStyle.Create;
begin
inherited;
FPenStyle:= psSolid;
FHighlightColor:= clBtnHighlight;
FShadowColor:= clBtnShadow;
end;

procedure Tem_LinuxConteinerVirtualization_CustomBoxStyle.SetPenStyle(Value:
TPenStyle);
begin  FPenStyle:= Value; if Assigned(OnChanged) then OnChanged(self); end;

procedure
Tem_LinuxConteinerVirtualization_CustomBoxStyle.SetHighlightColor(Value:
TColor);
begin  FHighlightColor:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

procedure Tem_LinuxConteinerVirtualization_CustomBoxStyle.SetShadowColor(Value:
TColor);
begin  FShadowColor:= Value; if Assigned(OnChanged) then OnChanged(self); end;

constructor Tem_LinuxConteinerVirtualization_CustomTextBoxStyle.Create;
begin
inherited;
FTextColor:= clBlack;
FBackgroundColor:= clWindow;
end;

procedure
Tem_LinuxConteinerVirtualization_CustomTextBoxStyle.SetTextColor(Value: TColor);
begin  FTextColor:= Value; if Assigned(OnChanged) then OnChanged(self); end;

procedure
Tem_LinuxConteinerVirtualization_CustomTextBoxStyle.SetBackgroundColor(Value:
TColor);
begin  FBackgroundColor:= Value; if Assigned(OnChanged) then OnChanged(self);
end;

constructor Tem_LinuxConteinerVirtualization_BevelLines.Create;
begin
inherited;
FStyle:= bvLowered;
FThickness:= 1;
end;

procedure Tem_LinuxConteinerVirtualization_BevelLines.SetCount(Value: cardinal);
begin FCount:= Value; if Assigned(OnChanged) then OnChanged(self); end;
procedure Tem_LinuxConteinerVirtualization_BevelLines.SetStep(Value: cardinal);
begin FStep:= Value; if Assigned(OnChanged) then OnChanged(self); end;
procedure Tem_LinuxConteinerVirtualization_BevelLines.SetOrigin(Value:
Tem_LinuxConteinerVirtualization_Origin);
begin FOrigin:= Value; if Assigned(OnChanged) then OnChanged(self); end;
procedure Tem_LinuxConteinerVirtualization_BevelLines.SetStyle(Value:
TPanelBevel);
begin FStyle:= Value; if Assigned(OnChanged) then OnChanged(self); end;
procedure Tem_LinuxConteinerVirtualization_BevelLines.SetBold(Value: boolean);
begin FBold:= Value; if Assigned(OnChanged) then OnChanged(self); end;
procedure Tem_LinuxConteinerVirtualization_BevelLines.SetThickness(Value: byte);
begin FThickness:= Value; if Assigned(OnChanged) then OnChanged(self); end;
procedure Tem_LinuxConteinerVirtualization_BevelLines.SetIgnoreBorder(Value:
boolean);
```

```
begin FIgnoreBorder:= Value; if Assigned(OnChanged) then OnChanged(self); end;

{ TGradient }

procedure TGradient.Draw(DC: HDC; r: TRect; PenStyle, PenWidth: integer);
var
i,j,x,y,x2,y2,h,w,NumberOfColors:integer;
c1F,c2F,c3F:byte;
c1T,c2T,c3T:byte;
c1D,c2D,c3D:integer;
_R,_G,_B:byte;
Pen,OldPen:HPen;
FillBrush:HBrush;
BufferBmp, OldBMP:HBITMAP;
BufferDC,TargetDC:HDC;
ColorR:TRect;
LOGBRUSH:TLOGBRUSH;

procedure SwapColors;
var TempColor: Longint;
begin
TempColor:= FRGBFromColor; FRGBFromColor:= FRGBToColor; FRGBToColor:= TempColor;
end;
begin
if (Steps = 1)or(GetDeviceCaps( DC, BITSPIXEL )<16) then
begin exit;
FillBrush:= CreateSolidBrush( ColorToRGB( FromColor ) );
FillRect( DC, r, FillBrush );
DeleteObject( FillBrush );
exit;
end;
x:=r.left; y:=r.top; h:=r.bottom-r.top; w:=r.right-r.left;
x2:= 0; y2:= 0; pen:= 0; oldpen:= 0; BufferDC:= 0;

if Orientation = fgdHorzConvergent then
begin
FOrientation:= fgdHorizontal;
Draw(DC, Rect(R.Left, R.Top, R.Right, R.Bottom- h div 2), PenStyle, PenWidth);
SwapColors;
Draw(DC, Rect(R.Left, R.Top+ h div 2, R.Right, R.Bottom), PenStyle, PenWidth);
SwapColors;
FOrientation:= fgdHorzConvergent;
exit;
end;
if Orientation = fgdVertConvergent then
begin
FOrientation:= fgdVertical;
Draw(DC, Rect(R.Left, R.Top, R.Right- w div 2, R.Bottom), PenStyle, PenWidth);
SwapColors;
Draw(DC, Rect(R.Left+ w div 2, R.Top, R.Right, R.Bottom), PenStyle, PenWidth);
SwapColors;
FOrientation:= fgdVertConvergent;
exit;
end;

c1F:=Byte(FRGBFromColor );
c2F:=Byte(WORD(FRGBFromColor) shr 8);
c3F:=Byte(FRGBFromColor shr 16 );
c1T:=Byte(FRGBToColor );
c2T:=Byte(WORD(FRGBToColor) shr 8);
c3T:=Byte(FRGBToColor shr 16 );
c1D:=c1T- c1F;
c2D:=c2T- c2F;
c3D:=c3T- c3F;

if BufferedDraw then
begin
BufferDC:= CreateCompatibleDC(DC);
```

```
BufferBmp:= CreateBitmap( w, h, GetDeviceCaps( DC, PLANES ), GetDeviceCaps( DC,
BITSPIXEL ), nil );
OldBMP:= SelectObject( BufferDC, BufferBmp );
SetMapMode( BufferDC, GetMapMode(DC) );
TargetDC:= BufferDC;
end else TargetDC:= DC;

case Orientation of
fgdHorizontal:
begin
NumberOfColors:= min( Steps, h );
ColorR.Left:= r.left; ColorR.Right:= r.right;
end;
fgdVertical:
begin
NumberOfColors:= min( Steps, w );
ColorR.Top:= r.top; ColorR.Bottom:= r.bottom;
end;
fgdLeftBias, fgdRightBias:
begin
NumberOfColors:= min( Steps, w+h );
if PenStyle=0 then PenStyle:=PS_SOLID; if PenWidth=0 then PenWidth:=1;
y2:= y;
if Orientation = fgdLeftBias then x2:= x else
begin x:= r.right; x2:= r.right; end;
end;
else{fgdRectanEm_LinuxConteinerVirtualization}
begin
h:=h div 2; w:=w div 2;
NumberOfColors:= min( Steps, min(w,h) );
end;
end;
LOGBRUSH.lbStyle:= BS_HATCHED;
LOGBRUSH.lbHatch:= Ord(BrushStyle)- Ord(bsHorizontal);
for i:=0 to NumberOfColors-1 do
begin
_R:= c1F+ MulDiv(i, c1D, NumberOfColors- 1);
_G:= c2F+ MulDiv(i, c2D, NumberOfColors- 1);
_B:= c3F+ MulDiv(i, c3D, NumberOfColors- 1);

case Orientation of
fgdHorizontal, fgdVertical, fgdRectanEm_LinuxConteinerVirtualization:
begin
if BrushStyle = bsSOLID then
FillBrush:= CreateSolidBrush( RGB( _R, _G, _B ) )
else
begin
LOGBRUSH.lbColor:= RGB( _R, _G, _B );
FillBrush:= CreateBrushIndirect(LOGBRUSH);
end;

case Orientation of
fgdHorizontal:
begin
if fReverse then begin
ColorR.Top:= r.bottom- MulDiv( i, h, NumberOfColors);
ColorR.Bottom:= r.bottom- MulDiv( i+ 1, h, NumberOfColors);
end else begin
ColorR.Top:= r.top+ MulDiv( i , h, NumberOfColors);
ColorR.Bottom:= r.top+ MulDiv( i+ 1, h, NumberOfColors);
end;
end;
fgdVertical:
begin
if fReverse then begin
ColorR.Left:= r.right- MulDiv( i,w, NumberOfColors);
ColorR.Right:= r.right- MulDiv( i+ 1, w, NumberOfColors);
end else begin
ColorR.Left:= r.left+ MulDiv( i,w, NumberOfColors);
```

```
ColorR.Right:= r.left+ MulDiv( i+ 1, w, NumberOfColors);
end;
end;
fgdRectanEm_LinuxConteinerVirtualization:
begin
ColorR.Top:= r.top+ MulDiv( i, h, NumberOfColors);
ColorR.Bottom:= r.bottom- MulDiv( i, h, NumberOfColors);
ColorR.Left:= r.left+ MulDiv( i, w, NumberOfColors);
ColorR.Right:= r.right - MulDiv( i, w, NumberOfColors);
end;
end;
FillRect( TargetDC, ColorR, FillBrush );
DeleteObject( FillBrush );
end;
else {fgdLeftBias, fgdRightBias:}
begin
if Pen <> 0 then
DeleteObject(SelectObject( TargetDC, OldPen ));
Pen:= CreatePen( PenStyle, PenWidth, RGB( _R, _G, _B ) );
OldPen:= SelectObject( TargetDC, Pen );
for j:= 1 to MulDiv( i+ 1, h+w, NumberOfColors)- MulDiv( i, h+w, NumberOfColors)
do
begin
case Orientation of
fgdLeftBias:
begin
if y >= r.bottom then inc( x, PenWidth ) else y:= y+ PenWidth;
if x2 >= r.right then inc( y2, PenWidth ) else x2:= x2+ PenWidth;
MoveToEx( TargetDC, x, y, nil );
LineTo( TargetDC, x2, y2 );
end;
else{fgdRightBias:}
begin
if x <= r.left then inc( y, PenWidth ) else x:= x- PenWidth;
if y2 >= r.bottom then dec( x2, PenWidth ) else y2:= y2+ PenWidth;
MoveToEx( TargetDC, x, y, nil );
LineTo( TargetDC, x2, y2 );
end;
end;
end;
DeleteObject( SelectObject( TargetDC, OldPen ) );
end;
end;
if NumberOfColors=0 then exit;
if i/NumberOfColors*100 > PercentFilling then break;
end;

if BufferedDraw then
begin
BitBlt( DC, 0, 0, r.right-r.left, r.bottom-r.top, BufferDC, 0, 0, SRCCOPY );
DeleteObject( SelectObject( BufferDC, OldBMP ) );
DeleteDC( BufferDC );
end;

end;

function Tem_LinuxConteinerVirtualization_Bevel.BordersHeight: integer;
begin
Result:= 0;
if Inner <> bvNone then
begin
if fsdTop in Sides then inc(Result);
if fsdBottom in Sides then
if Bold then inc(Result, 1) else inc(Result);
end;
if Outer <> bvNone then
begin
if fsdTop in Sides then inc(Result);
if fsdBottom in Sides then
```

```
if Bold then inc(Result, 1) else inc(Result);
end;
end;

function Tem_LinuxConteinerVirtualization_Bevel.BordersWidth: integer;
begin
Result:= 0;
if Inner <> bvNone then
begin
if fsdLeft in Sides then inc(Result);
if fsdRight in Sides then
if Bold then inc(Result, 1) else inc(Result);
end;
if Outer <> bvNone then
begin
if fsdLeft in Sides then inc(Result);
if fsdRight in Sides then
if Bold then inc(Result, 1) else inc(Result);
end;
end;

function Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle.HighlightColor:
TColor;
begin
Result:= incColor(Color, 60);
end;

function Tem_LinuxConteinerVirtualization_CustomListBoxItemStyle.ShadowColor:
TColor;
begin
Result:= decColor(Color, 60);
end;

end.
```