Довідка
про відсутність плагіату у випускній кваліфікаційній роботі

Міністерство освіти і науки України
Український державний університет науки та технологій

Кафедра «Комп'ютерні інформаційні технології»

ДОВІДКА

За результатами перевірки випускної кваліфікаційної роботи здобувача вищої освіти Білокрилець Лілії Олександрівни
(прізвище, ім'я, по батькові)
на тему: Дослідження часових характеристик затримки передавань в системах реального часу
в роботі не виявлено порушень академічної доброчесності.

Керівник ВКР _____ / Нечай В.Я /

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет науки і технологій

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»

Завідувач кафедри
_____ /Вадим ГОРЯЧКІН/

«_17_» _____12_____ 20_21_ р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань **12 Інформаційні технології**

Спеціальність **121 Інженерія програмного забезпечення**

Тема **Дослідження часових характеристик затримки переривань в системах реального часу**

Theme **Researching time characteristics of interrupt latency in realtime systems**

Керівник дипломної роботи доц. _____ Віктор НЕЧАЙ

Нормоконтролер доц. _____ Олена КУРОП'ЯТНИК

Студентка групи ПЗ2021 _____ Алла ВОЛКОДАВЕЦЬ

Student Alla VOLKODAVETS

Дніпро – 2021

Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Факультет Комп'ютерних технологій і систем кафедра Комп'ютерні інформаційні технології
Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ В. І. Шинкаренко
(підпис)
«19» листопада 2021 р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС магістр

студентки групи ПЗ2021 Волкодавець Алли Олександрівни

1 Тема дипломної роботи: Дослідження часових характеристик затримки переривань в системах реального часу

затверджена наказом по університету від «18» листопада 2020 р. № 690.

2 Термін подання студентом закінченої роботи «3» грудня 2021 р.

3 Вихідні дані до дипломної роботи _____
_____
_____
_____
_____

4 Зміст пояснювальної записки (перелік питань до розробки) огляд тематичної літератури та програмних аналогів; опис та обгрунтування методології замірів та аналізу часових показників затримки обробки переривань в операційній системі реального часу QNX; розробка програмного забезпечення автоматизації проведення досліджень часових характеристик; результати експериментальних замірів часових характеристик затримки обробки переривань та їх статистичного аналізу; вимоги охорони праці під час проведення аналогічних досліджень.

5 Перелік демонстраційного матеріалу презентація з результатами виконання роботи на тему «Дослідження часових характеристик затримки переривань у системах реального часу» та сформульованими висновками; відеозапис демонстрації роботи програмного комплексу автоматизації досліджень затримки обробки переривань; тези доповідей про результати дослідницької роботи на наукових конференціях.

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Техніко-економічні розрахунки	доцент Гненний М. В.		
Охорона праці	професор Саблін О. І.		

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Назва розділів дипломної роботи	Термін виконання розділів роботи	Примітка
1	Вступ	02.08.2021 - 15.08.2021	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	16.08.2021 - 29.08.2021	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	30.08.2021 - 05.09.2021	
4	Постановка задачі, технічне завдання	06.09.2021 - 12.09.2021	
5	Техніко-економічні показники	13.09.2021 - 19.09.2021	
6	Розробка інструментальних засобів дослідження	20.09.2021 - 17.10.2021	30%
7	Виконання досліджень	18.10.2021 - 31.10.2021	
8	Оформлення тез доповідей	01.11.2021 - 07.12.2021	
9	Оформлення статті у фаховий журнал	08.11.2021 - 14.11.2021	60%
10	Оформлення пояснювальної записки	15.11.2021 - 28.11.2021	
11	Розробка демонстраційних матеріалів	29.11.2021 - 05.12.2021	100%

Дата видачі завдання «18» листопада 2020 р.

Керівник дипломної роботи _____ В. Я. Нечай
(підпис) (ПІБ)

Завдання прийняв до виконання _____ А. О. Волкодавець

РЕФЕРАТ

Об‘єктом дослідження є затримка обробки переривань у системах реального часу.

Предметом дослідження є часові характеристики затримки обробки переривань та їх вплив на час виконання програм в операційній системі QNX.

Мета роботи — аналіз та оцінка впливу затримки переривань на час виконання їх обробки, а також на метрику найгіршого часу виконання програм у цілому.

Методи дослідження — проведення багатократних експериментальних замірів часу затримки; розрахунок значень математичного очікування, дисперсії, стандартного відхилення для отриманих результатів експериментів.

Дана пояснювальна записка складається з наступних розділів:

- вступ — опис суті дослідження, мети проведення дослідницької роботи, актуальність та наукової новизни. Складається з 6 сторінок;
- перший розділ — огляд наявних у відкритому доступі тематичних літературних джерел та програмних засобів. Складається з 12 сторінок;
- другий розділ — описує методологію проведення дослідження. Складається з 9 сторінок;
- третій розділ — описує процес проектування та розробки програмного інструментарію. Складається з 21 сторінки;
- четвертий розділ — опис експериментального аспекту даної дослідницької роботи з наведенням отриманих результатів. Складається з 21 сторінки;
- п‘ятий розділ — опис вимог безпеки праці, інформація про шкідливі фактори та можливі небезпечні ситуації. Складається з 10 сторінок;
- додатки — містять технічне завдання, робочий проект та тези, складені за результатами даної дослідницької роботи.

Таблиць — 34, рисунків — 22, бібліографія — 29 джерел.

Ключові слова: операційні системи реального часу, ОСРЧ, QNX, переривання, обробка переривань, WCET, статистичний аналіз, евристичний аналіз.

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

1. Переривання — повідомлення або сигнал, для обробки якого система повністю призупиняє свою роботу на деякий час.
2. Операційна система реального часу, ОСРЧ — система, що проектується для розв‘язку задач режиму реального часу.
3. QNX — UNIX-подібна операційна система реального часу, розроблена канадською компанією QNX Software Systems, Ltd.
4. Worst Case Execution Time, WCET — метрика найгіршого часу виконання програми, дозволяє визначити максимальний час, за який виконається та чи інша програма.
5. IRQ — Interrupt ReQuest, абревіатура на позначення ліній переривань — віртуальних каналів, через які система отримує сигнали від джерел переривань.
6. ISR — Interrupt Service Routine, функція обробки переривань, тобто, функція, що реагує на сигнал переривання і виконує певні дії щодо роботи з цим сигналом.
7. HDD — hard disk drive, привід жорсткого диску комп‘ютера.

ЗМІСТ

ВСТУП

Автоматизація керування фізичними процесами потребує розвитку засобів, що дозволяють звільнити людину від рутинної роботи. Більш того, «ручне» керування деякими процесами є фізично неможливим через їх швидкоплинність. На розв‘язок цих задач значним чином впливає стан розвитку галузей інформаційних технологій, пов‘язаних з керуванням фізичними процесами у реальному режимі часу. Саме цим обумовлена поява операційних систем реального режиму часу, які забезпечують розв‘язок задач у відведений фізичною системою час.

Автоматизація процесів тісно пов‘язана з таким поняттям, як задачі реального часу. Під задачами реального часу розуміються задачі, невчасне виконання яких може привести до небажаних наслідків, ступені впливу яких можуть варіюватися від незначних відхилень у фактичному графіку роботи до тотального краху усієї фізичної системи. За критичністю таких небажаних наслідків задачі реального часу можна розділити на такі категорії:

- задачі жорсткого реального часу — правильний розв‘язок даних задач повинен відбуватись не пізніше зазначеного часу;
- задачі м‘якого реального часу — отримання розв‘язку таких задач допускає незначні затримки у часі;
- задачі, некритичні до реального часу — задачі, вчасний розв‘язок яких не є обов‘язковим критерієм до організації процесу розв‘язку.

Задачі жорсткого (й інколи м‘якого) реального часу потребують особливої уваги, та, відповідно, особливих умов для їх розв‘язку. Саме з цією метою різними компаніями-розробниками ведуться роботи щодо розробки різних операційних систем реального часу. Їх головною метою є забезпечення своєчасності та коректності розв‘язку задач реального часу, покладених на сучасні системи керування фізичними об‘єктами (для таких систем часто використовується термін «вбудовані системи»), такі, як автотранспорт, медичне приладдя, робототехніка та ін. Операційна система QNX, розроблена компанією QNX Software Systems, Ltd. (з 2011 року — дочірнє підприємство канадської компанії BlackBerry), є популярною комерційною операційною системою реального часу (ОСРЧ), що успішно використовується у

вбудованих системах, тому важливо враховувати можливості та функціонал цієї ОСРЧ під час проектування власного програмного забезпечення для роботи з вбудованими системами.

Термін «переривання» говорить сам за себе — для обробки події, по відношенню до якої використовується це слово, система повинна на незначний час зупинити, або перервати, свою роботу. Зупинка роботи системи реального часу для обробки певної внутрішньої події може мати негативні наслідки, якщо не керувати її часовими характеристиками. Тому під час розробки програмних систем розв‘язку задач реального часу постає суттєве питання оцінки найгіршого часу роботи системи, представленого як однойменна метрика WCET (Worst Case Execution Time).

Однією з найбільш важливих вимог, що пред‘являються до ОС реального часу (ОСРЧ), є їх передбачуваність, тобто, можливість апріорі визначити величину WCET. Крім часу виконання коду програми, на цю величину мають вплив такі випадкові характеристики, як час виконання системних викликів, час роботи засобів синхронізації, час планування потоків до виконання, а також час затримки переривань, що можуть мати різне значення для різного типу задач. Тому визначення середньостатистичного значення цих характеристик дозволить уточнити величину WCET для конкретного класу задач.

Для переривань такі показники, як кількість ліній переривань та час на переключення контексту, варіюються в залежності від характеристик обраного процесору для конкретної системи, тому більшість досліджень та посібників з дослідження часових показників затримки переривань стосуються саме виконання замірів на рівні апаратури з використанням осцилографа. Актуальність даної роботи та наявності наукової новизни частково полягає в дослідженні переривань та затримки їх обробки саме з програмної точки зору. Незалежно від апаратури, системні виклики, що використовуються всередині ОСРЧ QNX, є однаковими для усіх розробників систем розв‘язку задач реального часу, тому при розробці програмного забезпечення, орієнтованого на коректність роботи, а також своєчасність отримання результатів та апаратну сумісність, слід враховувати значення затримки обробки переривань всередині програми та, якщо ці показники не відповідають очікуваним для розв‘язку необ-

хідної задачі, оптимізувати програмну систему, що розробляється, відповідно до поставлених часових вимог. Існуючі дослідження затримки переривань на програмному рівні, що будуть розглянуті в огляді існуючих наукових робіт та тематичної літератури, у різних версіях операційної системи надають чисельні значення затримки без детальних пояснень процедури їх отримання та контексту інтерпретування для використання під час розробок систем різного призначення. У результаті проведення даного дослідження розглянуті не лише окремі випадки виникнення затримки переривань, а й розроблений набір програмного інструментарію, який дозволить будь-якому користувачеві ОСРЧ QNX провести дослідження щодо затримки переривань у власних системах та для різних версій цієї операційної системи.

Об‘єктом дослідження є затримка обробки переривань — час між фіксуванням системою факту отримання сигналу про виникнення переривання та виконанням першої строки коду функції обробки цього сигналу у програмі відповідного призначення.

Предметом дослідження є часові характеристики затримки обробки сигналів переривань та їх вплив на метрику найгіршого часу виконання програм в операційній системі QNX.

Мета роботи — оцінити вплив затримки переривань на час виконання їх обробки всередині програм, розроблених для операційної системи реального часу QNX, за різних можливих умов, визначених вище, а також на вплив затримки переривань на метрику найгіршого часу виконання програм у цілому, що дозволить аналізувати такі програми на необхідність в оптимізації для поліпшення часових показників виконання.

Завдання роботи полягає у реалізації наступних пунктів, необхідних для досягнення мети:

- розробка низки програм для виконання замірів часу між отриманням сигналу про переривання системою та початком виконання функції обробки даного сигналу, за різних умов;
- фіксування результатів виконання даних програм за використання окремо віртуальної та реальної ЕОМ зі встановленою ОСРЧ QNX, а та-

кож порівняння результатів в обох випадках на предмет наявності впливових розбіжностей на час обробки;

- аналіз отриманих результатів статистичними методами та розробка власного програмного забезпечення з аналогічним функціоналом для дослідження часових характеристик обробки переривань в ОСРЧ QNX;
- аналіз та формулювання висновку на базі отриманих результатів щодо впливу затримки обробки переривань на загальну метрику WCET для програм вирішення задач жорстокого часу, розроблених для ОСРЧ QNX.

Методи дослідження — проведення багатократних експериментальних замірів часу між фіксацією системою сигналу про переривання та початком виконання функцій їх обробки; розрахунок значень математичного очікування, дисперсії, стандартного відхилення для отриманих результатів експерименту для визначення теоретичного значення часу затримки.

Дослідження найгіршого часу виконання програм на основі даних про часові характеристики затримки обробки переривань та інших окремих складових програмного забезпечення, що надаються конкретною операційною системою (наприклад, засоби синхронізації), є темою, що досі не є дослідженою та задокументованою у повному обсязі, особливо для операційних систем реального часу. Отримані знання про вплив часових характеристики затримки переривань на рівні програмного забезпечення та методи їх дослідження можливо використовувати як для проведення аналізу часових показників роботи програмного забезпечення обробки системних переривань, так і для доповнення аналізу метрики найгіршого часу виконання програм для ОСРЧ QNX існуючими методами, що використовують комп‘ютерну обробку експериментальних даних.

Оцінка та аналіз найгіршого часу виконання програми, що включає окремий аналіз затримки обробки переривань та використання його результатів для отримання загальних показників, може бути використаний розробниками програмних систем розв‘язку задач жорсткого часу, що використовують переривання як один з методів взаємодії з навколишнім середовищем (наприклад, через периферійні пристрої або

електронні датчики), для отримання інформації про те, чи зможе програмне забезпечення, що розробляється для такої системи, на предмет дослідження даних часових характеристик, забезпечити вчасну та точну реакцію системи на зміни у середовищі її функціонування, або ж чи потребує система оптимізації для забезпечення такої реакції. До систем, що можуть віднайти переваги у використанні програм для ОСРЧ QNX разом з розробленим у даній роботі програмним забезпеченням оцінки часу обробки переривань, можна віднести:

- транспортні системи комерційного та стратегічного призначення з функцією автопілоту (автомобілі, потяги, літаки, космічні кораблі, воєнна техніка);
- медичне обладнання, що використовує периферійні пристрої (ультразвукові діагностичні апарати, апарати життєзабезпечення, в т.ч. апарати штучного дихання, хірургічна робототехніка);
- системи автоматизації залізничного руху (системи контролю залізничної інфраструктури, системи HVAC (підігріву, вентиляції та охолодження) рухомого складу);
- системи контролю інфраструктури різних галузей промисловості (наприклад, видобутку енергії — атомні та гідроелектростанції).

Апробація результатів досліджень — процес дослідження та отримані результати описані у тезах, поданих до наступних конференцій:

- XIV Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (15 –16 грудня 2020 року, м. Дніпро);
- Всеукраїнська науково-технічна конференція молодих учених, магістрантів та студентів «Науково-технічний прогрес на транспорті» (29 березня 2021 року);
- 81 Всеукраїнська науково-технічна конференція молодих учених, магістрантів та студентів «Наука і сталий розвиток транспорту» (28 жовтня 2021 року);

– XV Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (16 –17 грудня 2021 року, м. Дніпро).

Також результати досліджень були оголошені на наукових семінарах кафедри КІТ за 22.02.2021 та 08.12.2021.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ДОСЛІДЖЕННЯ ЗА НАУКОВИМИ ЛІТЕРАТУРНИМИ ДЖЕРЕЛАМИ

1.1 Огляд тематичних наукових робіт та літератури

Розглянемо усі доступні на момент виконання даної роботи літературні джерела, що висвітлюють тему системних переривань та їх обробки, розділивши їх на три категорії, які будуть представлені далі.

1.1.1 Огляд офіційної документації до ОСРЧ QNX

Офіційне керівництво розробника QNX Neutrino присвячує перериванням та їх обробці окремий розділ, фокусуючись на питанні поняття переривання, системних викликів керування перериваннями та їх обробкою, налаштування контролеру PCI для визначення джерела переривання та чинники, які необхідно враховувати при розробці програм, включаючи чинники, що впливають на своєчасну обробку переривань. Аналізуючи документацію для більш пізніх версій QNX, а саме, 6.3.2 [1] та 7.1 [2], було помічено ідентичність викладу розділів та опису відповідних системних викликів, що свідчить про можливість адаптації програм обробки переривань під пізні версії.

1.1.2 Аналіз статей та наукових робіт, присвячених вивченню обробки переривань в ОСРЧ QNX

Однією з дослідницьких робіт щодо вивчення особливостей ОСРЧ QNX, є стаття Хільдебренда [3] «Архітектурний огляд QNX». Крім висвітлення переваг використання QNX версії 4.0, у кінці автором надаються експериментальні показники затримки переривань всередині ядра для процесорів з різними характеристиками.

Таблиця 1.1 — Дані щодо затримки переривань та процесів на різних моделях процесорів для QNX 4.0, нс

Процесор	Затримка переривання	Час обробки переривання	Затримка відкладеного переривання	Перемикання контексту
1	2	3	4	5
33 Mhz 486	6	5	14	17

Продовження таблиці 1.1

1	2	3	4	5
25 Mhz 486	8	7	18	22
33 Mhz 386	11	10	27	33
20 Mhz 386	19	17	45	55
16 Mhz 386SX	32	29	77	94
8 Mhz 286	65	59	163	188

Автор зазначає, що в даному випадку під найгіршим часом затримки переривань розуміється вказане у таблиці мінімальне значення затримки переривань, які виникають у системі, що не блокує лінії переривання на момент виникнення, плюс найдовший час блокування переривань системою або процесом. Інші нюанси отримання задокументованих даних в роботі Хільдебренда не вказані.

У відкритому доступі також наявні результати тестів продуктивності для QNX 6, проведені бельгійською компанією Dedicated Systems Experts [4], що спеціалізується на вивченні систем реального часу та проведенні незалежних досліджень різних часових показників їх роботи, за якими потім цим системам ставиться загальна оцінка продуктивності від 1 до 10. Для оцінки часових характеристик даної операційної системи, пов‘язаних з перериваннями та їх обробкою, компанією було проведено низку досліджень з використанням незалежних джерел переривань, а саме зовнішньої PCI-плати з системою на базі архітектури x86 та системного таймеру плати на базі архітектури ARM. Загалом для тестування було використано три ЕОМ на базі різних процесорів. Нижче наведені таблиці з результатами досліджень та отриманими значеннями середнього, максимального та мінімального часу затримки обробки переривань.

Таблиця 1.2 — Часові дані щодо затримки обробки переривань (час між переходом з виконуваного потоку до початку функції-обробника), отримані у результаті незалежних тестів Dedicated Systems Experts

Конфігурація ЕОМ	Середній час, мкс	Максимальний час, мкс	Мінімальний час, мкс
Комп‘ютер на базі процесору Pentium 200 MMX	1,8	5,8	1,7
Процесорний модуль Advantech SOM-6760 на базі процесору Intel Atom	1,7	2,8	1,7
Процесорна плата Beagle-XM Board ARM на базі процесору Cortex A8	0,5	2,6	0,5

Таблиця 1.3 — Часові дані щодо затримки перепланування (час між завершенням виконання функції обробки переривання та поверненням до потоку), отримані у результаті незалежних тестів Dedicated Systems Experts

Конфігурація ЕОМ	Середній час, мкс	Максимальний час, мкс	Мінімальний час, мкс
Комп‘ютер на базі процесору Pentium 200 MMX	1,4	13,1	1,4
Процесорний модуль Advantech SOM-6760 на базі процесору Intel Atom	1,2	3,4	1,1
Процесорна плата Beagle-XM Board ARM на базі процесору Cortex A8	0,6	2,6	0,5

Таблиця 1.4 — Часові дані щодо затримки переходу з обробника переривання у користувацький поток (час на розблокування потоку з найвищим проритетом, що включає затримку переключення контексту ядра між цими потоками), отримані у результаті незалежних тестів Dedicated Systems Experts

Конфігурація ЕОМ	Середній час, мкс	Максимальний час, мкс	Мінімальний час, мкс
Комп‘ютер на базі процесору Pentium 200 MMX	3,7	15	2,6
Процесорний модуль Advantech SOM-6760 на базі процесору Intel Atom	3,2	14,7	1,7
Процесорна плата Beagle-XM Board ARM на базі процесору Cortex A8	1,2	6	0,9

З отриманих результатів можливо побачити, що у даному випадку конфігурація ЕОМ, що тестувалася, цілком могла мати деякий вплив на результуючі часові дані. У статті, що наводить ці часові дані, не міститься інформація про методику проведення тестів, однак присутні порівняння отриманих значень з аналогічними характеристиками інших систем реального часу. Загалом, QNX Neutrino 6 отримала експертну оцінку продуктивності, що дорівнює 9/10 за умов тестування на конкретному апаратному забезпеченні.

1.1.3 Аналіз існуючих посібників та керівництв з теорії та практики обробки переривань в ОСРЧ QNX

Посібник Кртена «Введення в QNX Neutrino 2» [5] зрозуміло та з використанням життєвих прикладів викладає усі поняття та особливості архітектури ОСРЧ QNX, що описуються в офіційній документації до системи, включаючи переривання, яким присвячено окремий розділ. Автором надаються керівництво та приклади використання системних функцій для обробки переривань, описуються ситуації, що потребують відкладеної обробки переривань та використання кожного з системних

викликів роботи з перериваннями, а також надаються рекомендації щодо написання власних обробників переривань. Питання дослідження затримки переривань у даній книзі не розглядається.

Посібник Нікольського «Обробка переривань в операційній системі реального часу QNX» [6] з використання переривань у ОСРЧ QNX за своєю суттю є збірником лише релевантної до теми, що вивчається, інформації з документації до ОСРЧ QNX 6.2, розглянутого вище посібника Кртена та різних неофіційних джерел, таких, як російський форум qnx.org.ru, присвячений операційній системі, що також буде розглянутий у якості джерела. Проте цей посібник також містить результати авторських експериментів з замірів часових характеристик переривань за різних умов та у різних операційних системах реального часу, включаючи QNX версій 6.1 та 6.2. Ці результати наводяться нижче у мікросекундах.

Таблиця 1.5 — Час очікування обробки переривань IL та власне обробки на рівні Interrupt Service Routine (ISR) для QNX версій 6.1 та 6.2, мкс

	QNX 6.1	QNX 6.2
	середнє значення	макс. значення
IL	1,7	4,1
ISR	1,8	2,1

Також автором надається інформація про найменший інтервал часу Δ між періодично повторюваними перериваннями, за який операційна система встигає обробити усі переривання без їх втрати. Дані були отримані в результаті експерименту на часову оцінку очікування старту роботи функцій-обробників при великому навантаженні системи.

Таблиця 1.6 — Значення найменшого інтервалу часу Δ між періодично повторюваними перериваннями, мкс

ОС	Δ
QNX/Neutrino 6.1	10
QNX/Neutrino 6.2	9

Для виконання наступного експерименту автором була створена програма генерації мільярду переривань для підрахунку кількості оброблених та пропущених переривань.

Таблиця 1.7 — Затримка від виникнення переривання до початку його обробки функцією обробником для QNX 6.2, мкс

Пріоритети двох одночасних переривань	Значення затримки переривання		
	Мінімальне значення	Середнє значення	Максимальне значення
Високий (IRQ9)	1,6	1,6	2,5
Низький(IRQ10)	4,0	4,1	4,9

За результатами експериментів автор робить висновок про ефективність реалізації служби переривань у QNX 6.2, незважаючи на те, що значення для версії 6.2 є дещо більшими за значення для версії 6.1 [6].

1.2 Аналіз програмного забезпечення, аналогічного розробці

Для розгляду надаються аналогічні програмні інструменти, розроблені для різних UNIX-подібних операційних систем.

1.2.1 Тестування програми-аналога розробки cyclictest

cyclictest — текстова утиліта системного менеджеру операційної системи Debian [7], що входить до складу пакету прикладних програм realtime тестування характеристик затримок в ядрах Linux та Linux-rt (ядро для задач реального часу). Ця утиліта може використовуватись для часового аналізу затримки роботи потоків у будь-яких дистрибутивах Linux за різних параметрів, що задаються користувачем з командної строки. Деякі зі змінюваних параметрів включають вибір системного годинника (монотонного або реального часу), вибір функції фіксування значення часу затримки та визначення часу проведення експерименту, а також відображення числової гістограми отриманих значень. На рисунках нижче зображені скріншоти програми, отримані під час її запуску спеціально для виконання даного аналізу програмних аналогів з зазначеними параметрами у дії.

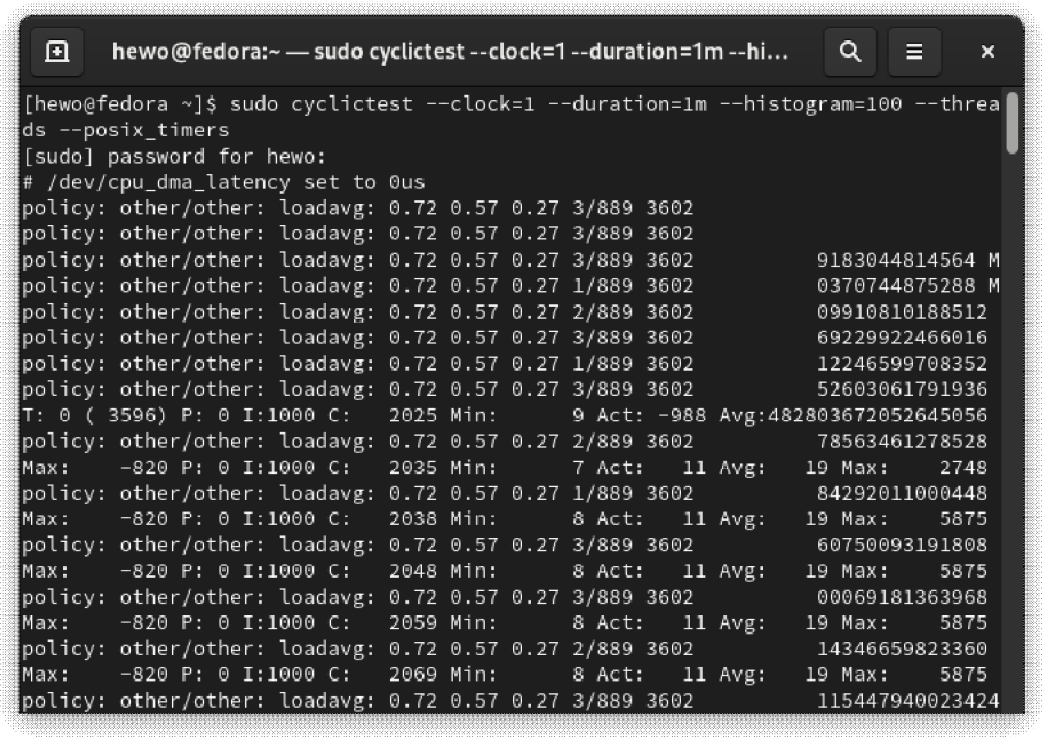

Рисунок 1.1 — Консоль операційної системи Fedora Workspace з завантаженою утилітою

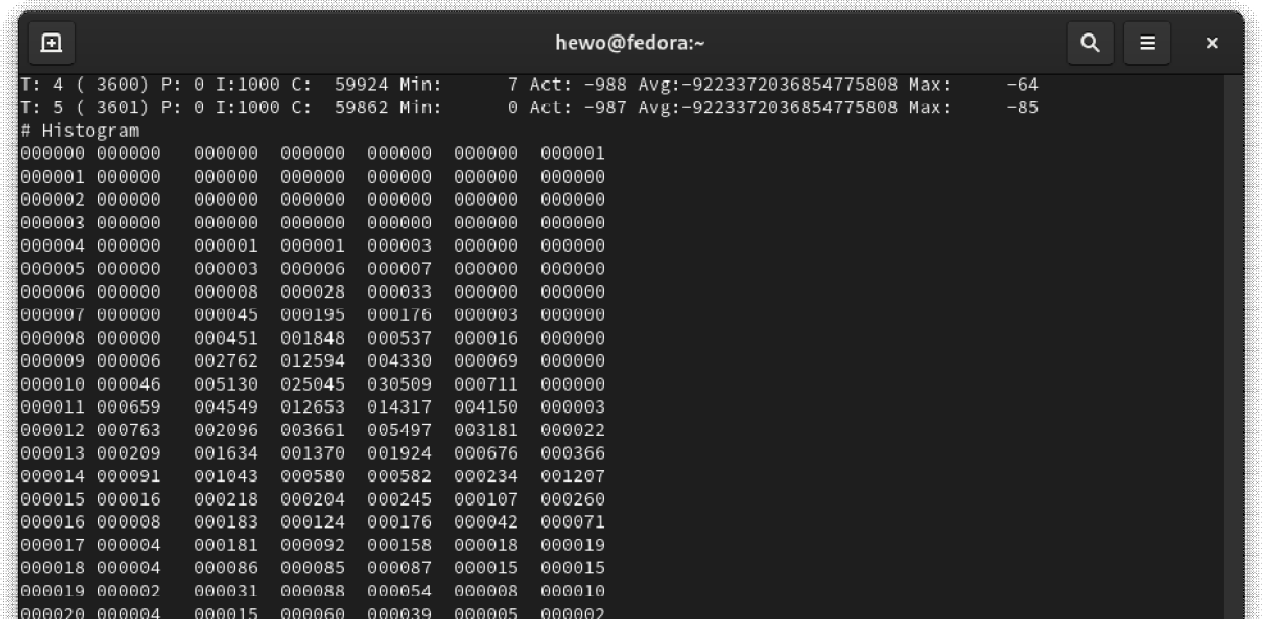

Рисунок 1.2 — Отримана за результатами вимірів гістограма затримки роботи

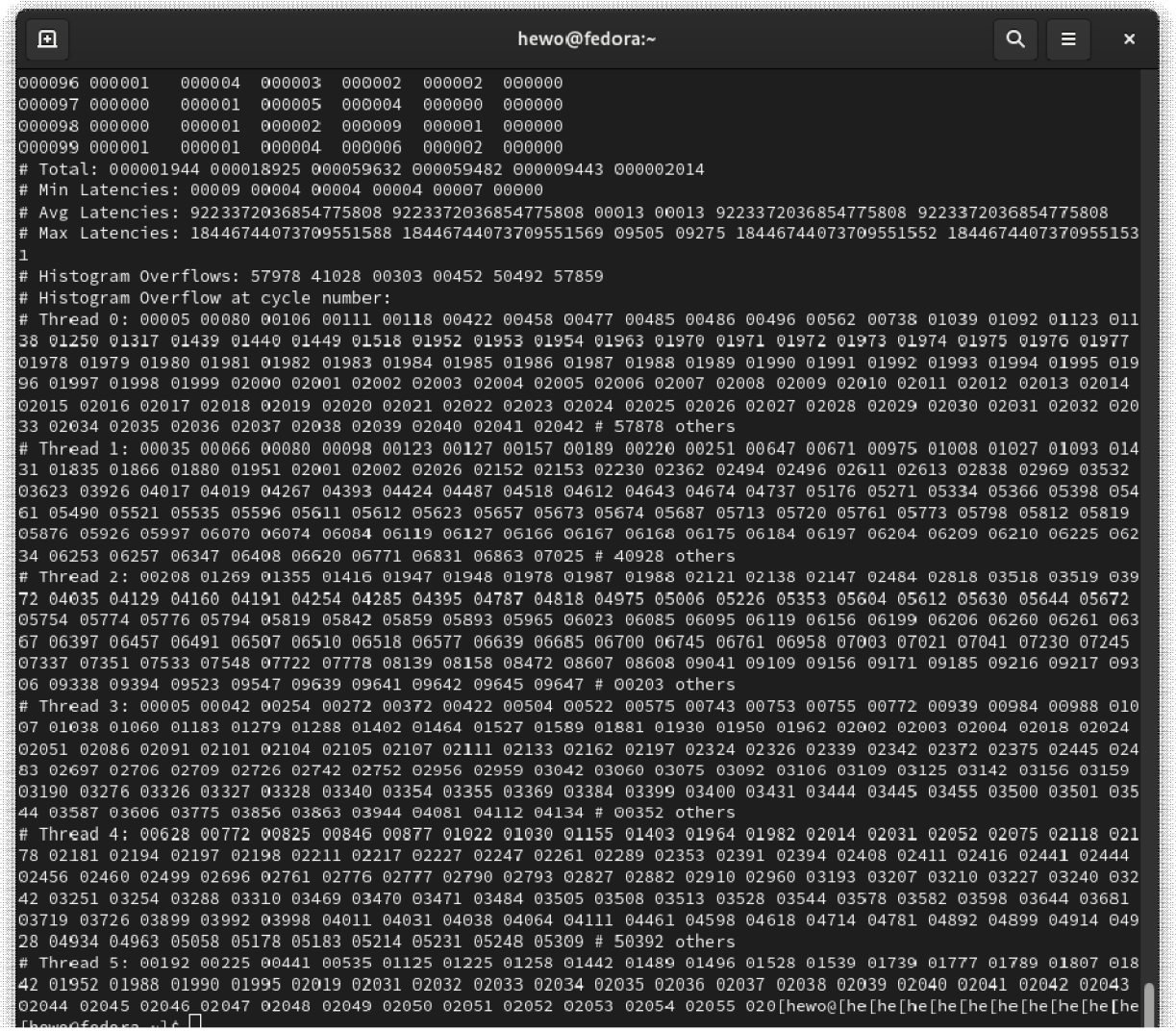

Рисунок 1.3 — Отримані статистичні результати обробки зафіксованих часових значень

1.2.2 Короткий огляд програмної утиліти gpioirqbench

gpioirqbench— утиліта, призначена для замірів затримки обробки зовнішніх переривань для інтерфейсів вводу/виводу загального призначення (GPIO) у ядрах фреймворку Xenomai, Linux та Linux-rt. Утиліта забезпечує можливість виконання замірів у цільових вбудованих системах через систему-хост таким чином: обидві системи підключаються одна до одної через сесію telnet, під час якої система-хост генерує переривання на контактах GPIO цільової системи та отримує від неї часові результати затримки у мікросекундах [8].

Лістинг 1.1 — Приклад можливих результатів роботи [8] утиліти

```
# gpioirq-latency
  period=10000us inner-loops=100
 ____Time _____Count ___Min ___Max ___Avg __Best _Worst ___Tos _Errors
  00:00:01        100      6     33      6      6     33      0   0   0
  00:00:02        200      6     31      7      6     33      0   0   0
  00:00:03        300      6     32      7      6     33      0   0   0
  00:00:04        400      6     31      7      6     33      0   0   0
  00:00:05        500      6     41      7      6     41      0   0   0
  00:00:06        600      6     34      7      6     41      0   0   0
  00:00:07        700      6     43      7      6     43      0   0   0
```

1.2.3 Дослідження існуючих засобів профілювання для ОСРЧ QNX

System Analysis Toolkit (SAT) для QNX — набір програмних інструментів профілювання роботи системи QNX, поставляється за умовчанням з комерційною або академічною ліцензією пакету QNX Momentics IDE[9]. SAT складається з наступних компонентів:

1. інструментоване системне ядро procnto-instr. На відміну від звичайного системного ядра procnto, інструментоване ядро є високоефективним у плані безперебійного збору інформації про ядро та усі процеси у системі, а також генерацію подій з фіксуванням їх часових характеристик та їх розміщенням у кільцевий зв‘язаний список буферів;
2. менеджер буферу ядра — цей буфер являє собою, як вже було сказано, зв‘язаний список великої кількості малих буферів, наповненість якого контролюється безпосередньо ядром, а також програмою фіксування даних tracelogger;
3. tracelogger — програма-деймон фіксування даних про події для їх подальшого запису або відправки іншим процесам для аналізу;
4. аналіз та інтерпретація даних — за допомогою утиліти traceprinter виводу відстежуваних подій, відсортованих за часом їх виходу з ядра, або бібліотеки libtraceparser, що надає широкий вибір функцій для обробки отриманих або зчитаних з «сирого» потоку даних буфером [9].

Усі вказані утиліти надаються разом з їх вихідним кодом, який можливо модифікувати відповідно до своїх потреб. Також інтегрована середа розробки QNX Momentics дозволяє відстежувати та профілювати події у ядрі операційної системи, що використовується для тестування розроблюваних в ній програм.

1.2.4 Розгляд апаратного рішення щодо експериментальних замірів

Осцилограф — прилад, що використовується для дослідження та отримання результатів роботи систем на рівні імпульсів електричного току та вважається найбільш ефективним засобом візуалізації та виконання замірів затримки переривань у часі завдяки майже повній відсутності погрішностей [9]. Зокрема, для замірів з їх візуалізацією використовуються цифрові осцилографи з дисплеєм.

1.3 Порівняльний аналіз розглянутих аналогів

Виконаємо порівняння розглянутих аналогів за допомогою таблиці, представленої нижче.

Таблиця 1.8 — Порівняння розглянутих методів за функціоналом, необхідним для реалізації програми для розв‘язку поставленої задачі

	cyclictest	gpioirqbench	SAT	осцилограф
Функціонал включає заміри затримки обробки переривань	+/-	+	+/-	+/-
Врахування програмних переривань	+/-	+/-	+	+/-
Сумісність з ОСРЧ QNX будь-яких версій	?	?	+	+
Автоматичний статистичний аналіз експериментальних даних	+	+	+/-	-
Візуалізація даних	+/-	-	?	+/-
Відкритий API/програмний код	+	+	+	-

Ключ:

- «+» — функціонал присутній;
- «+/-» — функціонал присутній частково або непрямо;
- «- »— функціонал відсутній;
- «?» — невідомо/не перевірялось.

Перерахуємо усі переваги та недоліки представленого програмного інструментарію для того, щоб визначити характеристики, необхідні для програмного забезпечення, що розробляється.

Таблиця 1.9 — Порівняння розглянутих методів за перевагами та недоліками використання

Метод	Переваги	Недоліки
1	2	3
cyclictest	Широка сумісність з існуючими дистрибутивами Linux (в теорії є можливість портування утиліти для QNX); можливість налаштування великої кількості параметрів для ефективної роботи	Виконання замірів затримки виконання потоків, а не лише потоків обробки переривань; текстовий інтерфейс, що ускладнює візуальну обробку експериментальних даних; відсутність прямої сумісності з QNX
gpioirqbench	Сумісність з існуючими дистрибутивами Linux; безпосереднє призначення утиліти — виконання замірів та аналізу затримки обробки переривань	Необхідність в наявності двох машин з портами GPIO для роботи з утилітою

Продовження таблиці 1.9

1	2	3
SAT	Повна сумісність з усіма підтримуваними на даний момент версіями QNX; широкий вибір функцій відкритого API для застосування у користувацьких додатках; детальна документація	Необхідність відповідних знань з програмування для ОСРЧ QNX для використання функцій API
осцилограф	Незалежність від програмної частини системи; наглядне представлення затримки обробки переривань а апаратному рівні; найбільш точні часові експериментальні показники	Необхідність у наявності та встановленні додаткового апаратного забезпечення для програмної підтримки даного методу; метод не враховує програмну структуру системи, що досліджується

Висновки за розділом 1

За результатами вивчення існуючих літературних джерел та програмних утиліт можна зробити висновок, що питання вивчення часових характеристик затримки переривань та їх впливу на роботу систем реального часу все ще є достатньо актуальним, однак методика проведення аналогічних експериментів, наглядне фіксування отриманих значень та аналіз результатів у середовищі ОСРЧ QNX більш пізніх версій в основному покладається на самого користувача за умови, що він має відповідні апаратно-технічні засоби та/або знання з програмування для UNIX-подібних операційних систем. Отже, цілком має місце створення програмного забезпечення для автоматизації аналізу затримки обробки програмних переривань за поставленим технічним завданням, з обов‘язковим урахуванням дослідженого у даному розділі існуючого досвіду дослідників та розробників ПЗ для систем реального часу.

2 ОБҐРУНТУВАННЯ МЕТОДІВ ДОСЛІДЖЕННЯ

2.1 Обґрунтування доцільності розробки

За результатами аналізу доступних для вивчення джерел було зроблено висновок про недостатність актуального тематичного матеріалу з досліджень та методології проведення часових замірів затримок обробок переривань, що у свою чергу, підтвердило доцільність виконання даного дослідження. Окрім того, програмний комплекс, що буде розроблений у результаті проведення дослідницької роботи, націлений на використання для різних версій операційної системи QNX, що дозволило проводити дослідження часу затримок за різних умов програмного та апаратного середовища.

2.2 Обґрунтування методології експериментальних замірів часу

Служба часу QNX складається з низки системних викликів та функцій, призначених для отримання часових даних у конкретний момент виконання програми, включаючи стандартні функції POSIX. Однак для отримання найбільш точних даних розробниками ОСРЧ QNX рекомендується використання системного виклику ClockCycles() фіксування поточного значення часу системного таймеру у циклах [1, 2], тому усі експерименти будуть проводитися з використанням часових даних, отриманих у результаті виклику ClockCycles(). Евристичне дослідження часових характеристик полягає у виконанні наступних дій:

- фіксація проміжних значень системного таймеру у циклах за допомогою функції ClockCycles() у конкретній ділянці програмного коду;
- знаходження різниці між отриманими значеннями часу у циклах для визначення тривалості коду, розташованого між командами заміру часу;
- конвертація отриманого значення у мікросекунди за допомогою макросу SYSPAGE_ENTRY отримання інформації про поточний процесор (значення кількості циклів таймеру на 1 секунду знаходиться за записом SYSPAGE_ENTRY(qtime)->cycles_per_sec).

2.2 Методи обробки переривань та їх дослідження у ОСРЧ QNX

Переривання — системний сигнал, що може використовуватись як засіб комунікацій між потоками та процесами, а також між апаратними складовими. Під обробником переривання (Interrupt Service Routine, ISR) у даному випадку розуміється користувацька функція, що починає свою роботу після отримання сигналу про виникнення конкретного переривання у системі. При цьому система бере на себе відповідальність за такі деталі, як переключення контексту з поточної функції до функції обробника. Час реакції системи на переривання, що включає інтервал часу між створенням відповідною апаратною частиною сигналу переривання та початком обробки цього сигналу відповідною функцією, залежить від низки факторів середовища, у якому відбувається фіксування та обробка сигналів про переривання — наприклад, характеристики операційної системи, ефективність та архітектура центрального процесору та оперативної пам‘яті і т.д. [5]. ОСРЧ QNX надає перелік власних функцій для маніпулювання сигналами переривань, які будуть розглянуті у даному розділі.

Відстеження програмних переривань для проведення експериментальних замірів відбувається за допомогою функцій прив‘язування переривань до потоків, а саме InterruptAttach() та InterruptAttachEvent(). Ці функції призначені для однакових задач, але мають дещо різний набір параметрів і тому мають свої переваги щодо використання у різних ситуаціях.

Функція InterruptAttach() має наступний прототип:

int InterruptAttach(int intr,const struct sigevent * (* handler)(void *, int), const void *area, int size, unsigned flags) [11]

У якості параметру, крім інших, InterruptAttach() приймає вказівник на користувацьку функцію-обробник переривання (зі вказівником на структуру події struct sigevent* у якості типу вихідних даних), що спрацює за призначенням при реєстрації сигналу виникнення переривання.

Крім функцій, переривання можуть прив‘язуватись до системних подій, що разом з сигналами та повідомленнями в ОСРЧ QNX використовуються як засоби

комунікацій між процесами та потоками. Для прив‘язування переривання, що обробляється, до структури sigevent на позначення події використовується функція InterruptAttachEvent(). Наведемо прототип даної функції нижче.

int InterruptAttachEvent(int intr, const struct sigevent*event, unsigned flags) [11]

InterruptAttachEvent() приймає у якості параметру вказівник на константний екземпляр структури sigevent, яка використовується як один з засобів комунікації потоків у середовищі QNX, тому при виникненні переривання потоком буде створена ця подія, що потім може бути передана іншим взаємодіючим потокам.

Для наглядного відображення відмінностей у функціонуванні системних викликів прив‘язування переривання потоки керування для обох функцій схематично представлені на рисунках нижче.

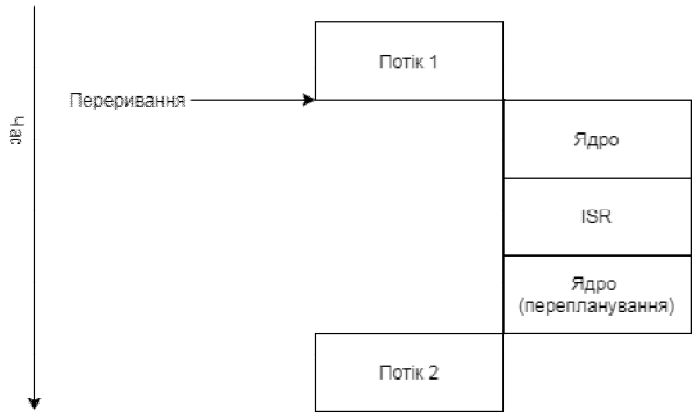

Рисунок 2.1 — Потік керування за використання виклику InterruptAttach()

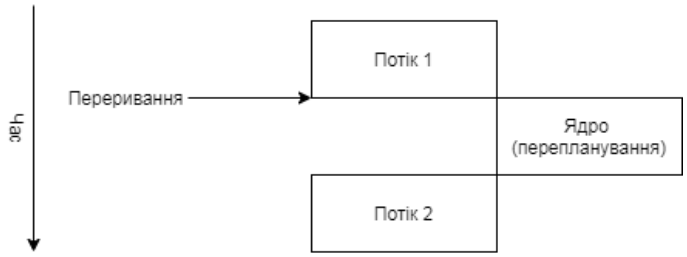

Рисунок 2.2 — Потік керування за використання виклику InterruptAttachEvent() [5]

Як показано на схемах вище, різниця між використанням даних функцій полягає у автоматичному переключенні контекстів потоку та функції-обробника (ISR на схемі) для InterruptAttach() та відсутності переключень для InterruptAttachEvent(). Використання InterruptAttachEvent() значно зменшує час, що витрачається системою

на обробку переривань, однак не завжди потрібно, щоб потік, до якого прив‘язане переривання, реагував на кожне його виникнення, тому за необхідності фільтрування сигналів про переривання більш доцільним є використання InterruptAttach() [5]. На відміну від використання функцій для обробки переривань, обробка на основі події не потребує від ядра додаткових дій щодо переключення контексту, однак програмісту необхідно власноруч демаскувати лінію переривання для можливості її подальшого відстеження потоками. Якщо маскування переривань полягає у тимчасовому ігноруванні контролером PIC сигналів про конкретне переривання, то демаскування лінії переривання означає повернення можливості контролеру реагувати на сигнали переривань. Для маскування та демаскування, або відновлення доступу до ліній переривань використовуються виклики InterruptMask() та InterruptUnmask() відповідно, які разом приймають у якості єдиного параметру ідентифікатор id прив‘язки лінії переривання до потоку. Задача механізму маскування полягає саме у наданні можливості винесення роботи з обробки апаратного переривання за межі потоку обробки для гарантованого отримання результуючих даних.

Для коректної роботи потоку виклику обробника після завершення обробки переривання необхідно звільнити цей потік від прив‘язаної до нього лінії переривання. Це можливо здійснити за допомогою системного виклику InterruptDetach(), що у якості єдиного параметру приймає значення ідентифікатора обробника переривання id, отриманого як вихідні дані системного виклику InterruptAttach() або InterruptAttachEvent().

За очікування прийому сигналу про настання переривання відповідає функція InterruptWait(), що використовується в обох випадках та має наступний прототип:

int InterruptWait(int flags, const uint64_t *timeout) [11]

Через те, що функції обробки переривань виконуються не на рівні користувача, а на рівні ядра, сама функція обробки повинна виконуватись за найменший можливий час, а перелік допустимих для використання всередині обробника функцій є обмеженим. Для кожної функції та системного виклику, інформація про яку надається в офіційній документації до ОСРЧ QNX [11], присутні позначки щодо

безпечності її використання в обробниках переривань, тому за необхідності використання конкретної функції або системного виклику в обробці конкретного сигналу переривання рекомендується у першу чергу звертатися до актуальної документації з бібліотечних функцій QNX. Кртеном надається перелік функцій, що є безпечними для використання в функціях обробки переривань, а саме:

- функції сімейства atomic_*(), призначені для елементарних математичних операцій, що виконуються за найменший можливий час;
- функції сімейства mem*() операцій над ділянками пам‘яті, визначених у стандартній бібліотеці функцій мови С;
- більшість функцій сімейства str*() операцій над символьними строками (у якості прикладу винятку автором надається функція strdup() дублювання строк, що використовує небезпечний у контексті потоків ядра системний виклик malloc() розподілу пам‘яті під дані);
- системні виклики маніпулювання доступом до ліній переривань — Inter-ruptMask(), InterruptUnmask(), InterruptLock(), InterruptUnlock(), InterruptDisable(), InterruptEnable();
- функції in*() та out*() виконання побітових операцій [5].

Також для QNX6 та пізніших версій операційної системи є безпечним для використання системний виклик ClockCycles() фіксування поточного системного часу у тактах системного таймеру [15], що був використаний для отримання часових значень затримки обробки переривань.

2.3 Випадки затримки обробки переривань

Обробка програмних переривань у даній роботі була досліджена для трьох конкретних випадків виникнення затримки, які було розглянуто у вступній частині, а саме:

- затримка планування обробки;
- затримка обробки відкладеного переривання;
- затримка обробки за пріоритетністю переривань.

Під затримкою планування, або диспетчеризації, розуміється затримка переключення з контексту користувацького потоку до контексту потоку ядра, що вико-

нує обробку переривання, тобто, спрацьовування коду обробки переривання одразу після виникнення. Як було розглянуто вище, в залежності від обраної функції прив‘язування лінії переривання до потоку час, витрачений на планування ядром потоку обробки переривання може варіюватись.

Відкладене переривання (deferred interrupt) полягає в тому, що потік-обробник переривання не виконується одразу, а тільки після його планування потоком, до якого прив‘язане переривання. Цього можливо досягти за допомогою планування потоку-обробника з використанням екземпляру структури події sigevent, що є одним з методів комунікацій потоків та процесів між собою у середовищі ОСРЧ QNX. Для використання події її атрибути завчасно заповнюються даними про функцію, що повинна запуститись у новому потоці. Найбільш простим та ефективним способом ініціалізації події для створення нового потоку є використання макросу SIGEV_THREAD_INIT. Прототип макросу виглядає наступним чином:

SIGEV_THREAD_INIT(struct sigevent &event, void(*sigev_notify_function) fn, union sigval value, pthread_attr * attr) [11]

У якості параметрів макрос створення потоку приймає вказівник на структуру події створення потоку event, вказівник на функцію fn для запуску у потоці, значення параметрів value для передачі функції та екземпляр структури атрибутів потоку attr. Кртен рекомендує використовувати генерацію потоків з використанням даної структури з обережністю [5], оскільки ініціалізація події може призвести до великої кількості незаблокованих та конфліктуючих між собою потоків.

Затримка за пріоритетністю переривань виникає у разі, якщо обробка переривання витісняється обробкою інших, більш пріоритетних переривань. Переривання самі по собі є задачами, обслуговування яких в ОСРЧ QNX є задачею високого пріоритету. Крім того, кожна лінія переривань в залежності від свого призначення або підключеного пристрою також має власний пріоритет. Цілком можлива ситуація, за якої потоки обробки переривань, що мають нижчий з доступних пріоритетів, можуть витіснятися обробниками переривань на більш пріоритетних лініях.

Кожен розглянутий вище випадок потребує окремого вивчення для отримання більш точних результатів, які можуть використовуватися для часового аналізу алгоритмів з функціоналом обробки переривань з різних джерел, з різним призначенням та за різних умов та налаштування середовища їх функціонування.

2.4 Використані методи статистичного аналізу даних

Усі часові характеристики, що отримуються експериментальним чином, мають випадковий характер, тому для їх аналізу у даному досліджені використовуються статистичні методи дослідження, що включають розрахунок набору статистичних метрик. Статистичні дані повинні бути відомими наприкінці проведення досліджень часових характеристик, оскільки таким чином можливо узагальнити велику кількість отриманих експериментальних даних для використання у розрахунках теоретичного значення найгіршого часу виконання програм WCET, однією з функцій яких є обробка переривань. Часові значення, що отримуються під час проведення даних досліджень, за визначенням Гмурмана [13] є дискретними випадковими величинами, тобто, величинами, можливі значення яких є окремі ізольовані числа (між двома отриманими сусідніми значеннями відсутні будь-які інші значення величини). Далі будуть розглянуті поняття, через які можливо охарактеризувати результуючі дискретні випадкові величини.

Математичне очікування M_ξ випадкової величини ξ за визначенням є сума добутків усіх її можливих значень на їх ймовірності [12]. Для зліченої дискретної випадкової величини, кожне значення якої є різним (або значення не згруповані), значення математичного очікування розраховується за наступною формулою [14]:

$$M_\xi = \frac{1}{n}\sum_{k=1}^{n} x_k p_k. \tag{2.1}$$

Математичне очікування є числовою характеристикою середнього значення випадкової величини [13], що може використовуватись замість експериментальних

даних при проведенні певних розрахунків, пов‘язаних з вимірюваною величиною. Математичне очікування також називається випадковим середнім [14].

До характеристик розсіювання можливих значень випадкової величини навколо математичного очікування відносять такі метрики, як дисперсія та середньоквадратичне, або стандартне, відхилення.

Під дисперсією D_ξ випадкової величини ξ розуміється математичне сподівання квадрату відхилення випадкової величини від її математичного сподівання [12], тобто,

$$D_\xi = M\left\{\left(\xi - M_\xi\right)^2\right\}. \tag{2.2}$$

Стандартне відхилення σ_ξ випадкової величини ξ — число, що дорівнює кореню зі значення дисперсії[12].

$$\sigma_\xi = \sqrt{D_\xi}. \tag{2.3}$$

Стандартне відхилення є мірою надійності значення математичного очікування та позначає, наскільки отриманні випадкові значення відхилені від отриманого значення математичного очікування. Чим менше квадратичне відхилення, тим краще математичне очікування відображає всю сукупність даних, що відображається [14].

Висновки за розділом 2

У даному розділі були розглянуті усі аспекти, що стосуються проведення досліджень затримки обробки переривань в ОСРЧ QNX евристичним та статистичним методами. Розглянуті системні виклики, що використовуються для написання тестових програм заміру часових характеристик, є викликами, що описані виключно у бібліотеці стандартних функцій QNX, що свідчить про зручність та нескладність написання програм, до функціоналу яких входить обробка переривань. Служба часу,

що для даного дослідження представлена лише одним системним викликом, також не повинна викликати труднощів при використанні. Незважаючи на помітну недостатність літературних джерел, що розкривають тему методології дослідження часу затримок переривань, наявність відповідних знань та досвіду роботи з операційними системами реального часу та з обробкою статистичних даних дозволила визначити усі необхідні для проведення дослідження засоби та формули, які були застосовані під час розробки програмного забезпечення автоматизації розрахунку часових характеристик обробки переривань у рамках проведення даного дослідження.

3 РОЗРОБКА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ ДОСЛІДЖЕННЯ ЧАСОВИХ ХАРАКТЕРИСТИК ЗАТРИМКИ ОБРОБКИ ПЕРЕРИВАНЬ В ОСРЧ QNX 6

Дане дослідження має на меті окремий аналіз кожної з розглянутих у попередньому пункті ситуацій виникнення затримок обробки переривань, тому інструментальні засоби, що входять до складу розробленого у результаті роботи програмного комплексу також будуть створюватися окремо для моделювання елементарних випадків кожної з трьох розглянутих ситуацій виникнення затримки обробки переривань у часі. Проектування інструментальних засобів буде виконуватись за допомогою UML-діаграм.

Unified Modeling Language, або UML — уніфікована мова візуального моделювання систем, у т.ч. програмного забезпечення. UML можливо використовувати для візуального моделювання за будь-яких обраних методологій проектування. Моделі — сукупність сутностей проекту та взаємодії між ними — візуально представляються за допомогою діаграм, яких в UML описується 13 різних видів, з власними позначками та призначенням [19]. Далі будуть розглянуті деякі види діаграм, що використовуються для візуального представлення програмних засобів, що розробляються, та їх внутрішньої взаємодії.

3.1 Формалізація задачі

Формалізація задачі з розробки тематичного комплексу програмного інструментарію представлена у вигляді діаграми прецедентів. Діаграма прецедентів, або варіантів використання, використовується для визначення предметної області системи [26]. Для програмного забезпечення у діаграмах прецедентів у еліпсах вказуються опції використання програми, що можуть виконуватись користувачем (у даному випадку він називається актором та позначається на діаграмі як чоловічок). Позначка <<include>> використовується для дій, що включаються у більш загальну дію. Позначка <<extend>> використовується на позначення дій, якими інші дії можуть доповнюватися.

Діаграма формалізації задачі представлена на рис. 3.1 та містить короткий опис дій, які зможе виконувати користувач при безпосередній взаємодії з виконавчим файлом програми.

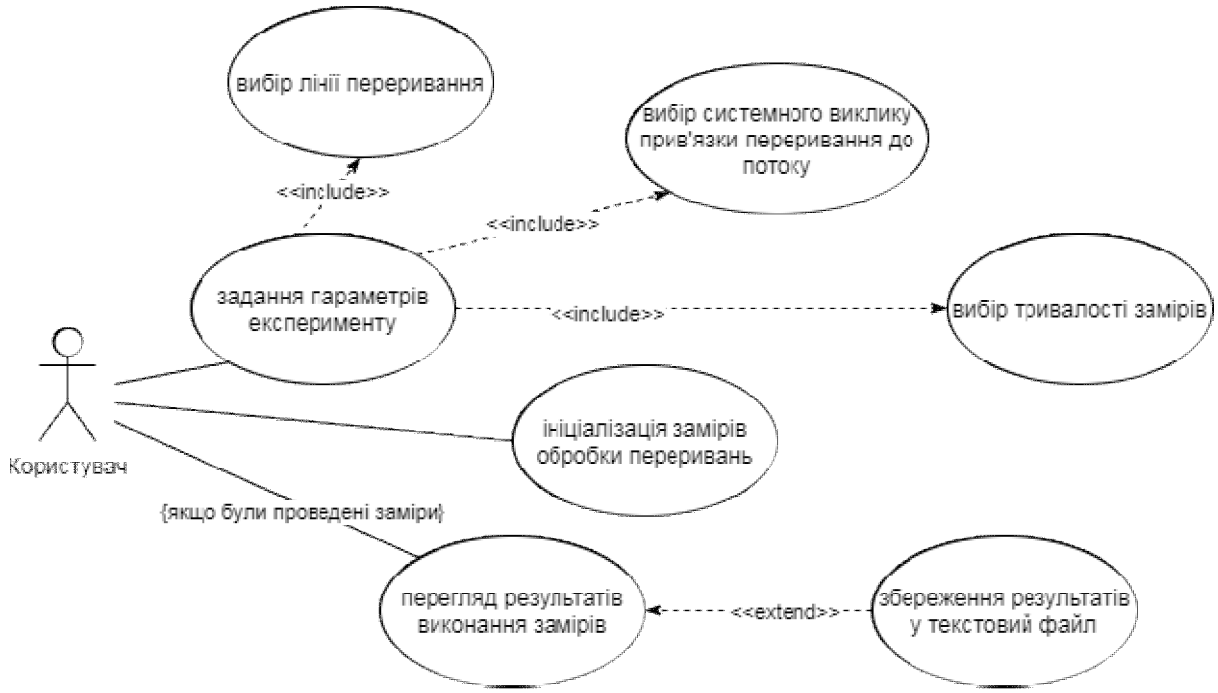

Рисунок 3.1 — Діаграма варіантів використання

3.2 Вибір мови програмування та середовища розробки

ОСРЧ QNX 6 дозволяє написання програм взаємодії з системою мовами C, C++ та Java. Для розробки програмного комплексу аналізу часу затримки переривань було обрано мову C++ на базі наступних критеріїв:

- сумісність з усіма бібліотечними функціями ОСРЧ QNX, що перераховуються в офіційній документації до операційної системи, а також POSIX-сумісними системними викликами, бібліотеками функцій та типів даних мови C;
- простота у визначенні складних структур даних (як-то класів) та виконання операцій роботи з текстовими та файловими даними завдяки власним бібліотекам типів даних та функцій;
- наявність власного попереднього досвіду розробки мовою C++ взагалі та конкретно для ОСРЧ QNX.

Відповідно до обраної мови для програмування та поставленої задачі була обрана об‘єктно-орієнтована та імперативна парадигма розробки програмного комплексу. Імперативна парадигма заснована на командах оновлення деяких змінних та полягає у використанні таких поширених у мовах програмування понять, як підпрограми, вирази, декларації та ін. Об‘єктно-орієнтована парадигма є галуззю імперати-

вного програмування, яка припускає використання виключно класів та об‘єктів класів, а також пов‘язаних з ними принципів, таких, як інкапсуляція, поліморфізм та наслідування [16]. Для проведення експериментів з замірів часу затримки обробки переривань будуть розроблені програмні засоби з використанням імперативної парадигми, а для об‘єднання усіх розроблених у якості результатів модулів в один програмний комплекс вже будуть застосовані поняття класів, об‘єктів та інкапсуляції.

3.2.1 Особливості мови C++ для використання у розробці

Мова C++ є об‘єктно-орієнтованою мовою програмування, тому її доцільно використовувати для реалізації програми за обраними парадигмами програмування. Відповідно до імперативної парадигми, C++ може використовуватись для написання програм, складених з деякого числа функцій, або підпрограм, що взаємодіють зі змінними. Разом з цим у мові C++ визначена можливість створення та опису класів та шаблонів класів, і це дозволяє розширити можливості використання програмного коду за межі простих функцій, наприклад, шляхом створення інтерфейсу розробки програм (API), на базі якого інші програмісти зможуть створювати власні утиліти автоматизації досліджень часу для обробників переривань з використанням вже існуючого коду.

Бібліотека стандартних функцій мови програмування C++ містить низку визначень структур, класів та функцій, які були визначені достатньо корисними для впровадження у програмний комплекс, що розробляється впродовж проведення даної дослідницької роботи. Результати замірів часу затримок обробки переривань у мікросекундах зберігаються у екземплярах контейнера типу вектор — std::vector. Вектор є контейнером зі змінним розміром для різних типів даних, опис якого включає велику кількість допоміжних методів з запису та зчитування з вектора, зокрема, функції доступу, схожі на визначення черги.

Для виводу інформації на консоль та у файл також використовуються стандартні типи даних бібліотек iostream та fstream відповідно, що значно спрощує реалізацію файлових операцій та текстового інтерфейсу користувача.

3.2.2 Особливості архітектури QNX Neutrino

Архітектура QNX є заснованою на пріоритетах та попереджуючою, або такою, що захвачує (preemptive). Це означає, що кожному потоку надається пріоритет, на основі якого їх надається доступ до ядра. Якщо одночасно створюються та запускаються потоки з високим та низьким пріоритетом, першим спрацює той потік, у якого пріоритет є вищим. Також архітектура QNX дозволяє мати місце наступній ситуації — якщо під час роботи потоку з низьким пріоритетом бажає спрацювати потік вищого пріоритету, ніж пріоритет робочого потоку, то потік з вищим пріоритетом захвачує процесор, тим самим перериваючи роботу потоку низького пріоритету. Саме ця ситуація і описується словом «preemptive».

Існує декілька способів планування, або диспетчеризації, потоків:

- FIFO — планування за принципом черги, «першим зайшов, першим вийшов»;
- Sporadic — планування зі встановленим лімітом часу на функціонування потоку на певному проміжку часу, корисне для аперіодичних подій;
- Round-robin — планування типу карусель, потоку виділяється відрізок часу timeslice, на протязі якого йому дозволено працювати, по завершенні цього проміжку часу потік припиняє роботу;
- Other — невизначений спосіб планування, значення якого різниться в залежності від обраної версії QNX, для QNX 6 це значення дорівнює Sporadic [4].

Потоки можуть мати пріоритет від 0 (найнижчого) до 255 (найвищого) незалежно від типу їх диспетчеризації, або планування. Для потоків, що не створюються від імені root, доступні номера пріоритетів знаходяться у діапазоні 1-63. У системі завжди функціонує потік idle, або фоновий потік, який має пріоритет 0 та завжди готовий до роботи. Обробники переривань мають вищий за будь-які інші потоки пріоритет, однак вони плануються у інший спосіб. Якщо системою зафіксовано сигнал переривання, тоді робочий на даний час потік втрачає управління процесором для захватом його обробника переривання, після чого апаратна частина запускає ядро, яке й планує відповідний потік обробки переривання, що виникло [1].

3.2.3 Середовище програмування для QNX Neutrino

Однією з переваг програмування для ОСРЧ QNX є наявність власних засобів розробки програмного забезпечення, що постачаються разом з ліцензійною копією самої операційної системи реального часу та є крос-платформним. QNX Momentics IDE — спеціалізоване середовище розробки з графічним інтерфейсом, яке дозволяє проводити розробку та тестування програм без потреби встановлення самого середовища в операційній системі. Це середовище розробки надає можливості віддаленого тестування програм, тобто, можливе написання самої програми на операційній системі за вибором користувача (Microsoft Windows, Linux, OS Solaris) та її тестування шляхом встановлення з‘єднання з іншою ЕОМ зі встановленою ОСРЧ QNX через спеціальний мережевий протокол qconn, ексклюзивний для операційної системи QNX.

3.3 Опис дослідницьких модулів

Для розробки програмного комплексу з дослідження часу обробки переривань була проведена декомпозиція початкового завдання з розробки на три модулі відповідно до трьох існуючих випадків затримки обробки переривань у ОСРЧ QNX.

Діаграма діяльності зображає системи вузлів, що з‘єднані ребрами. Вузли бувають трьох категорій — вузли дій, що представляють одиниці роботи (зображені прямокутниками з округленими кутами, можуть доповнюватися іншою символікою); вузли керування потоком діяльності; вузли об‘єктів, що використовуються у діяльності. Вузли керування можуть бути наступними:

- початковий та кінцевий вузли — зображені як зафарбовані круги, розташовані на початку та кінці діаграми відповідно. Кінцевий вузол додатково має зовнішнє коло;
- вузол рішення — вузол умовного переходу, зображується як ромб;
- вузол розгалуження та об‘єднання — використовується для позначення одночасних операцій, позначається як довгий зафарбований прямокутник [26].

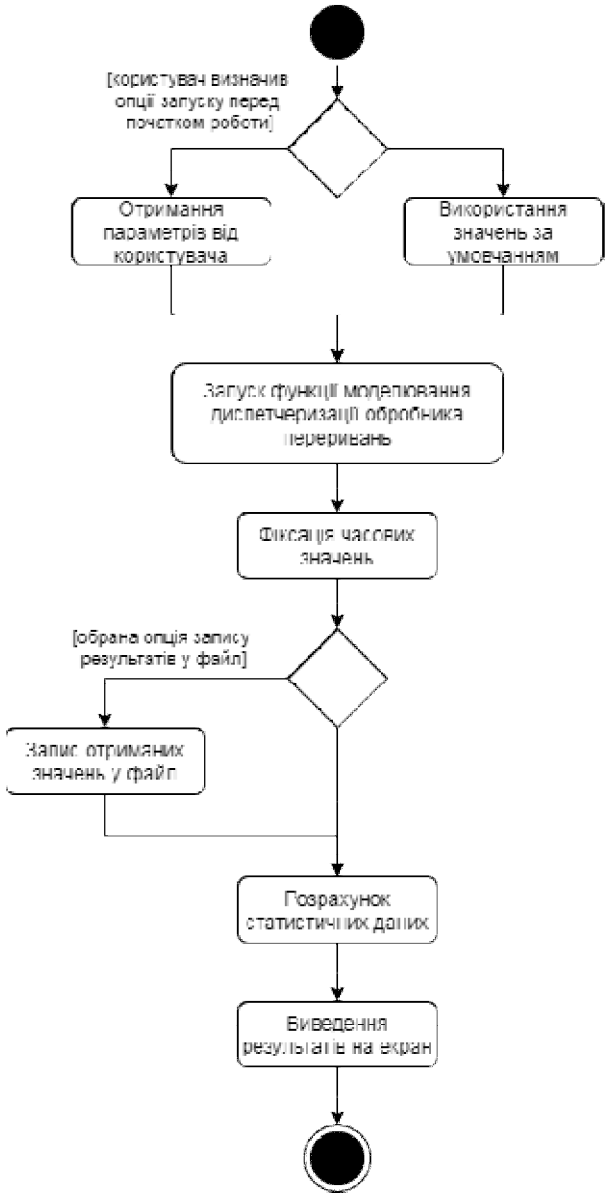

Рисунок 3.2 — Діаграма діяльності програм з моделювання та дослідження обробки переривань

Усі програмні модулі, що є складовими даного програмного комплексу, та зв‘язки між ними зображені на діаграмі артефактів нижче (рис. 3.3). Діаграма артефактів показує усі програмні артефакти (наприклад, виконавчі файли або файли налаштувань), що використовуються у проекті, і зв‘язки між ними.

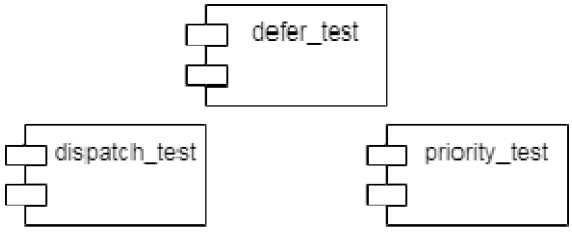

Рисунок 3.3 — Модулі програмного комплексу «QNX ISR Test Toolkit»

Представлені на діаграмі модулі є наступними:

- dispatch_test — модуль заміру часу затримки планування обробки;
- defer_test — модуль замірів затримки відкладеної обробки переривань;
- priority_test — модуль дослідження обробки переривань за пріоритетністю.

3.3.1 Модуль заміру часу затримки планування

Найпершим було розроблено модуль моделювання затримки планування обробки переривань та виконання замірів часу цієї затримки. Початкову версію цього модулю було використано передусім для перевірки наступних умов середовища дослідження обробки переривань:

- порівняння результатів часових замірів для «чистої» QNX та для віртуальної машини;
- порівняння результатів планування за нормального функціонування системного середовища та при його навантаженні у великих об‘ємах.

В залежності від обраної функції прив‘язування переривання до робочого потоку алгоритм планування обробника цього переривання має деякі відмінності. Для того, щоб прив‘язати лінію переривання до потоку, перед безпосереднім викликом будь-якої функції прив‘язування необхідно надати відповідні привілеї потокові на введення та виведення даних за допомогою системного виклику ThreadCtl() контролю потоку, якому потрібно передати параметр _NTO_TCTL_IO, та провести ініціалізацію структури події типу sigevent, що буде реагувати на сигнал, згенерований перериванням. На рисунку нижче представлена блок-схема алгоритму обробки переривання функцією-обробником (ISR), що було прив‘язане за допомогою функції InterruptAttach().

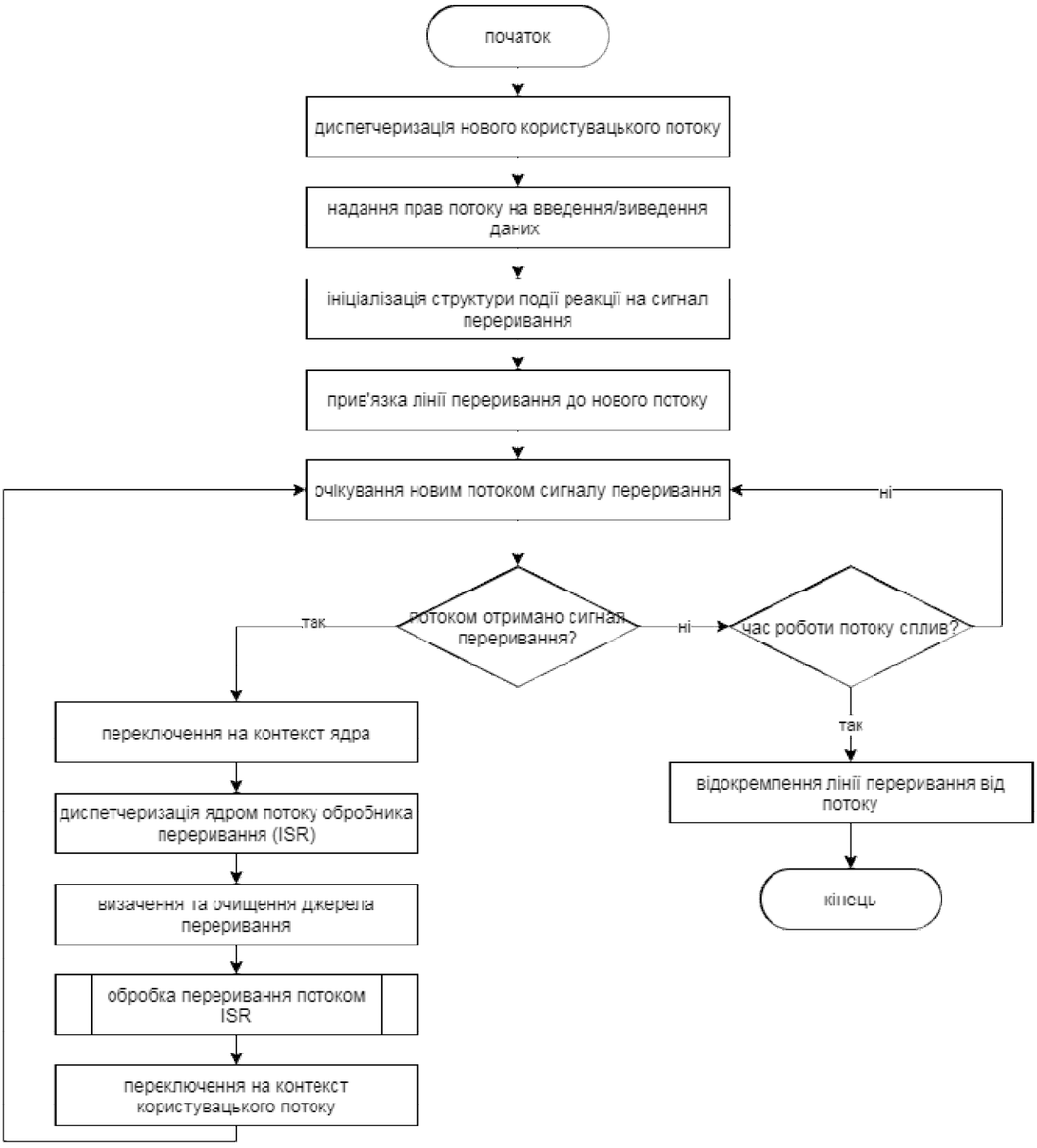

Рисунок 3.4 — Алгоритм обробки переривань з використанням ISR

Кртеном надається псевдокод типової функції ISR, що можливо реалізувати для будь-якої ОСРЧ, що містить описи системних викликів роботи з перериваннями та подіями. Псевдокод типового ISR [5] виглядає наступним чином:

FUNCTION ISR BEGIN

визначення джерела переривання

очищення джерела переривання

IF потрібно передати роботу потоку THEN

RETURN(подія);

ELSE

RETURN(NULL);

ENDIF

END

Код розробленого модулю:

```
//функція обробки переривання
//вхідні параметри: *area - ; id
//вихідні параметри: NULL
const struct sigevent *handler(void *area, int id)
{
    cycle2 = ClockCycles( ); //фіксація часу у циклах системного таймеру
    return(&event);
}
//функція потоку прив'язки переривання
//вхідний параметр: *arg - вказівник на параметри потоку
void *irq_thread(void *arg)
{
    int id;//ідентифікатор прив'язаного потоку
    std::ofstream log;//файловий потік для запису даних
    char logname[20];//ім'я файлу
    //відкриваємо файл для запису результатів, якщо встановлено відповідний прапор
    if (file_io_flag)
    {
        itoa(irq, logname, 10);
        strcat(logname, ".log");
        log.open(logname, ios::out | ios::app);
    }
    if (ThreadCtl( _NTO_TCTL_IO, NULL ) < 0) }//надання прав потоку на введення-виведення
    {
        std::cout << "ThreadCtl returned an error: "<< strerror(errno);
        return EXIT_FAILURE;
    }
    event.sigev_notify = SIGEV_INTR;//ініціалізація події для використання обробником
    //прив'язка до переривання в залежності від обраного методу:
    id = using_event
        ? InterruptAttachEvent(irq, &event, 0)//події
        : InterruptAttach(irq, &handler, NULL, 0, 0);//функції handler
    //цикл очікування сигналу перериванняif (id < 0 ) //перевірка на неправильну прив'язку лінії переривання
    {
        std::cout << "InterruptAttach";
```

```
                if (using_event) std::cout << "Event";
                std::cout << "() returned an error: " << strerror(errno);
                return EXIT_FAILURE;
            }
            while (1)
            {
                cycle1 = ClockCycles( );
                InterruptWait(0, NULL);//функція очікування, що спрацює одразу після отримання сигналу переривання
                if (using_event)
                {
                    cycle2 = ClockCycles( );

        InterruptUnmask(irq, id);//якщо до переривання прив'язана подія, демаскуємо лінію переривання
                }
                //запис отриманого значення часу у файл
                if (file_io_flag)
                {
                    log << (double)(cycle2 - cycle1)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000) << std::endl;
                }
                //запис отриманих значень часу у відповідні вектори

        results_cycles.push_back(cycle2 - cycle1);

        results.push_back((double)(cycle2 - cycle1)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000));
            }
            //закриття файлового потоку
            if (file_io_flag)
            {
                log.close();
            }
            //від'єднання переривання від потоку
            InterruptDetach(id);
        }
```

Процедура прив‘язки переривання до екземпляру структури події є аналогічною до прив‘язки функції, однак переключення контексту ядра у даному випадку не відбувається. Натомість, програміст повинен самостійно визначити, якими чином подія, ініціалізація якої була проведена у результаті отримання робочим потоком сигналу про настання переривання, буде використана для обробки переривання, зазвичай, з використанням нового користувацького потоку, диспетчеризація якого відбувається паралельно робочому потоку. Для програми моделювання затримки планування настання події через спрощення реалізації не обробляється ніяким чином, однак у реальних програмних умовах цю особливість необхідно враховувати та ви-

користовувати з обережністю, аби не перезавантажувати систему надлишковою обробкою постійно виникаючих переривань, прив‘язаних до певної події. На рис. 3.5 зображено алгоритм обробки переривань вищезазначеним методом.

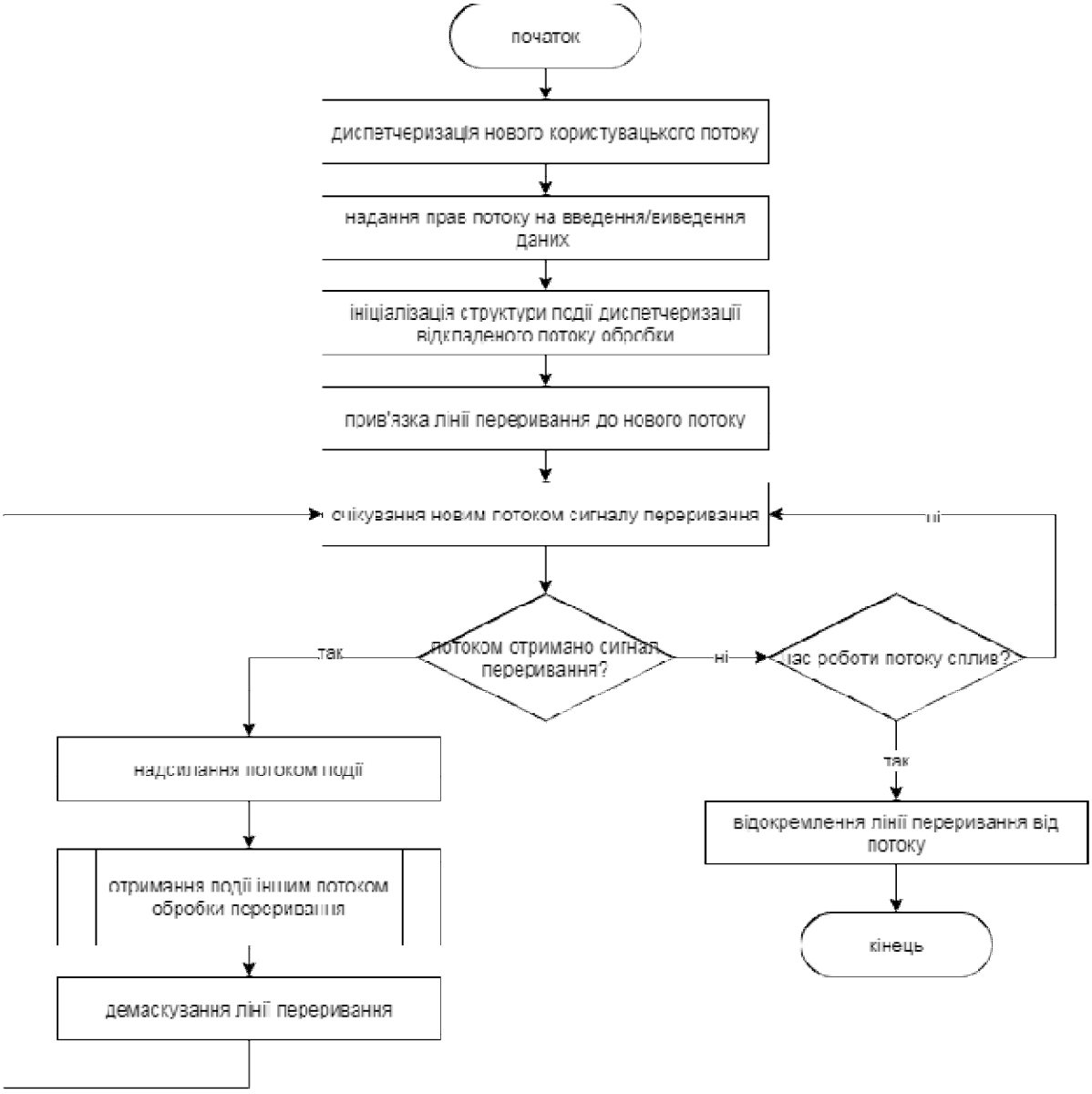

Рисунок 3.5 — Алгоритм обробки переривань з використанням подій

3.3.2 Модуль виконання замірів затримки відкладеної обробки переривань

Відкладена обробка передбачає наявність окремої функції, що буде запускатись за сигналом-подією, отриманим від потоку ядра, який був згенерований конкретним сигналом переривання. Для цього, як вже було відмічено, використовується макрос SIGEV_THREAD_INIT, перед викликом якого встановлюються необхідні права робочого потоку та атрибути події, за допомогою якої системою буде заплановано новий потік обробки, як у випадку з простим плануванням.

Алгоритм виконання відкладеної обробки переривань представлений нижче на рис. 3.6.

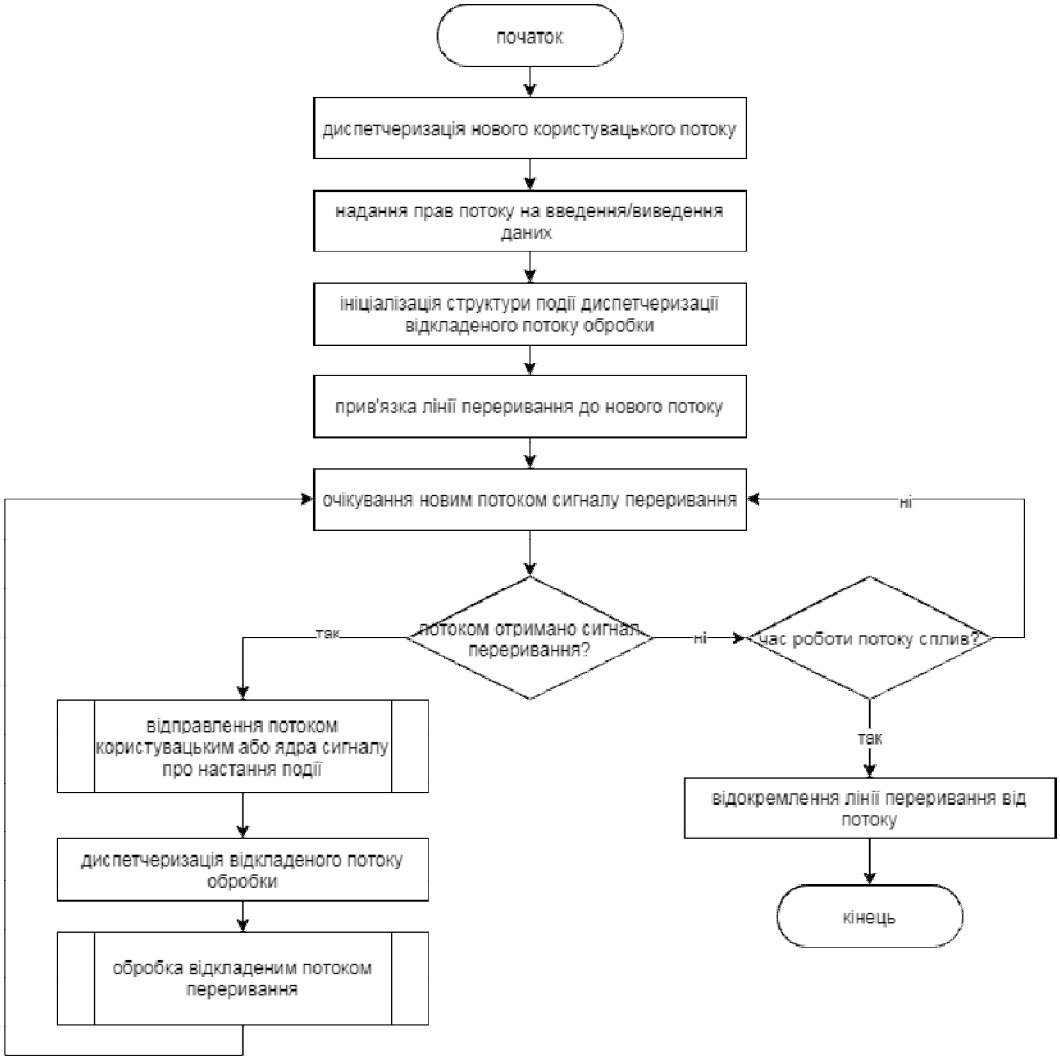

Рисунок 3.6 — Алгоритм відкладеної обробки переривань

Планування потоку з використанням подій можливо завдяки модифікації відповідних атрибутів екземпляру структури sigevent. Для налаштування екземпляру структури події на створення потоку у разі отримання програмою сигналу про появу події необхідно визначити відповідні атрибути з використанням макросу SIGEV_THREAD_INIT.

Код розробленого модулю моделювання затримки відкладеної обробки:

```
//функція відкладеної обробки пере-
ривання
//вхідний параметр: weh - значення
отриманого сигналу від потоку ядра
void isr_thread(union sigval weh)
{
    cycle2 = ClockCycles();
    if (file_io_flag)
    {
        logger
```

```
                    << (double)(cycle2 - cycle1)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000) << std::endl;
                }
                results_cycles.push_back((uint64_t)cycle2 - cycle1);
                results.push_back((double)(cycle2 - cycle1)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000));
                pthread_exit(NULL);
        }
        //функція-потік ядра відкладеного планування потоку-обробки
        const sigevent *deferred_thread(void *area, int size)
        {
                cycle1 = ClockCycles();
                return (&defer_event);
        }

        void *irq_thread(void *arg)
        {
                int id;
                if (ThreadCtl( _NTO_TCTL_IO, NULL ) < 0) }//надання прав потоку на введення-виведення
                {
                        std::cout << "ThreadCtl returned an error: "<< strerror(errno);
                        return EXIT_FAILURE;
                }

                id = InterruptAttach(irq, &deferred_thread, NULL, 0, 0);
                if (id < 0 ) //перевірка на неправильну прив'язку лінії переривання
                {
                        std::cout << "InterruptAttach() returned an error: " << strerror(errno);
                        return EXIT_FAILURE;
                }
                SIGEV_THREAD_INIT(&defer_event, &isr_thread, NULL, NULL);

                while (1)
                {
                        InterruptWait(0, NULL);
                }
                InterruptDetach(id);
        }
```

3.3.3 Модуль дослідження обробки переривань за пріоритетністю

Потоки, що функціонують у межах одного процесу, розділяють між собою змінні поза стеком викликів, а також обробники сигналів, канали та з‘єднання, тому у разі необхідності одночасна обробка потоками одного процесу різних переривань може бути організована програмістом, у тому числі зі спільним використанням структур подій, функцій обробки тощо.

Сама по собі обробка переривань є завданням з високим пріоритетом, яке виконується системою одним з перших у черзі. Проте усі доступні лінії переривання мають власне значення пріоритету, тому важливо дослідити часові значення затримки обробки переривань різного пріоритету за умови, що програма, призначена для обробки переривань, може оброблювати переривання на декількох лініях одночасно.

Даний модуль моделює ситуацію, при якій під час роботи потоку, до якого прив‘язана лінія переривання з низьким пріоритетом, головним потоком процесу-контейнеру виконується планування та запуск потоку з прив‘язаним до нього перериванням вищого пріоритету. Метою моделювання та дослідження такої ситуації полягає у розрахунку часу, за який виконується потік ядра, що виконує обробку переривання з нижчим пріоритетом. Припускається, що обробка переривання, нижчого за пріоритетністю, відбувається довше, якщо паралельно ядром плануються потоки обробки переривань з вищою пріоритетністю. Для порівняння часу виконання обробки програмою спочатку запускається потік обробки переривання з низьким пріоритетом та фіксується час, після чого паралельно запускається потік з прив‘язаною до нього лінією переривання системного таймеру, сигнал від якого обробляється за допомогою функції, тобто, відбувається прив‘язка переривання за допомогою InterruptAttach(). Схематично алгоритм роботи відображений на рис. 3.7.

Рисунок 3.7 — Алгоритм моделювання ситуації затримки обробки переривання за пріоритетністю

Код розробленого модулю затримки обробки за пріоритетами:

```
//функція-обробник переривання
const sigevent *isr_thread2(void *area, int size)
{
    //заміри часу виконання функції
    cycle1 = ClockCycles();
    atomic_add(&dummy_value, 1); //інкрементування змінної dummy_value безпечною функцією
    cycle2 = ClockCycles();
    return (&defer_event); //обробник повертає подію для повідомлення основного потоку про своє завершення
}
//функція-обробник переривання без замірів часу
const sigevent *isr_thread3(void *area, int size)
{
    cycle3 = ClockCycles();
    atomic_sub(&dummy_value, 1);
    cycle4 = ClockCycles();
    return (&defer_event);
}
//функція потоку, що реагує на переривання вищого пріоритету (переривання системного таймеру)
void *hiprio_thread(void *arg)
{
    int id;
    if (ThreadCtl( _NTO_TCTL_IO, NULL ) < 0) }//запит на надання привілегій потоку на приєднання обробника переривань
    {
        std::cout << "ThreadCtl returned an error: "<< strerror(errno);
        return EXIT_FAILURE;
    }
    SIGEV_INTR_INIT(&defer_event); //ініціалізація екземпляру структури події для використання
    id = InterruptAttach(0, &isr_thread3, NULL, 0, 0); //приєднуємо до даного потоку обробник переривання системного таймеру (на лінії IRQ0)
    if (id < 0 ) //перевірка на неправильну прив'язку лінії переривання
    {
        std::cout << "InterruptAttach() returned an error: " << strerror(errno);
        return EXIT_FAILURE;
    }

    while (1)
    {
        InterruptWait(0, NULL);//поток циклічно очікує повідомлення про сигнал переривання

        time_vector2.push_back
        ((double)(cycle4 - cycle3)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000));

    }
    InterruptDetach(id);//від'єднання переривання від даного потоку
```

```
        pthread_exit(NULL);//завершення роботи потоку
    }
    //функція потоку, що реагує на переривання нижчого пріоритету (переривання контролеру жорсткого диску)
    void *loprio_thread(void *arg)
    {
        int id;
        double result;
        ThreadCtl( _NTO_TCTL_IO, 0 );//запит на привілегії прив'язування переривання
        SIGEV_INTR_INIT(&defer_event);//ініціалізація події
        id = InterruptAttach(irq, &isr_thread2, NULL, 0, 0);//прив'язка переривання та його обробника до потоку
        while (1)
        {
            InterruptWait(0, NULL);//поток циклічно очікує повідомлення про сигнал переривання
            result = (double)(cycle2 - cycle1)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000);
            if (!fancy_flag)
            {
                //якщо прапор не активований (має значення брехні), потік виконується у нормальному режимі, записуємо часові дані в перший вектор
                //if (time_vector.back() != result)

                time_vector.push_back (result);
            }
            else
            {
                //якщо прапор активований (має значення істини), то паралельно виконується потік з перериванням вищого пріоритету, запис часових даних в другий вектор

                prio_vector.push_back (result);
                if (file_io_flag)
                {
                    logger << prio_vector.back() << std::endl;
                }
            }
        }
        InterruptDetach(id); //від'єдання переривання
        pthread_exit(NULL);//завершення потоку
    }
```

Результати проведення експериментів наведені у розділі 4 даної пояснювальної записки.

3.4 Тестування програми

Об‘єм програмних модулів, створених у результаті проведення досліджень, є досить невеликим для того, щоб застосовувати комплексні методи тестування та налагодження. Крім того, особливості роботи обробки переривань у ОСРЧ QNX не дає можливості налагодити функції-обробники програмними засобами. Середовище розробки QNX Momentics містить вбудовані опції запуску програми для налагодження та профілювання, що дають можливість дещо спростити процедуру тестування програм, які функціонують у середовищі ОСРЧ QNX.

Тестування коректності роботи програмних модулів було проведено наступними методами модульного тестування (unit testing), а саме методами чорного та білого ящику.

Методи модульного тестування націлені на перевірку окремих програмних модулів, що ідеально підходить для програмного комплексу, який розробляється у ході виконання дослідницької роботи.

Методи чорного ящику полягають у тестуванні програми без попередніх знань про її внутрішню будову. Інколи цей метод також називають функціональним. Методи білого, або прозорого, ящику, на відміну від методу чорного ящика, використовуються при знаннях апріорі про внутрішній склад програми[20].

З існуючих методів чорного ящику для тестування даної програми був використаний метод припущення про помилку. Цей метод означає дедуктивне визначення причини виникнення помилки, відповідно до якого програма налагоджується та перевіряється на коректність роботи. Через складність налагодження функції обробки переривань наявність відповідних знань про архітектуру ОСРЧ QNX та поняття обробки переривань, а також досвіду програмування, значним чином може допомогти у розв‘язку проблем за умов неможливості відстеження їх джерела.

Наведемо список використаних методів білого ящику:

- покриття коду — перевірка, чи виконуються усі строки коду;
- покриття умов — перевірка, чи виконуються умовні блоки коду за різних значень вхідних даних.

Таблиця 3.1 — План тестування програми

№ з/п	Що перевіряється	Вхідні дані	Очікувані вихідні дані
1	Коректність зчитування параметрів командної строки для заповнення відповідних змінних	Параметри командної строки з помилками	Повідомлення про помилку
		Параметри командної строки без помилок	Результати розрахунків
2	Запис отриманих часових значень у екземпляри контейнеру std::vector	Параметри командної строки	Коректні результати розрахунків за даними у екземплярах контейнеру std::vector
3	Перевірка коректності переключення контекстів потоків користувача та ядра	—	Коректні результати розрахунків за даними у екземплярах контейнеру std::vector

Перевірка програм за планом описана у підпунктах нижче.

3.4.1 Тестування коректності зчитування параметрів командної строки

Даний тест має на меті перевірку коректного зчитування програмою параметрів командної строки за допомогою функції getopt() та оператора switch налаштування умов зчитування отриманих параметрів. Фрагмент коду, що виконує дану функцію та є однаковим для усіх модулів моделювання затримки оброки переривань, наведемо нижче.

```
while((c = getopt(argc, argv, "i:efd:w:")) != -1 ) //цикл зчитування та обробки введених параметрів командної строки
    {
        switch(c)
        {
            case 'i':
                irq = atoi(optarg);
                break;
            case 'e':
                using_event = true;
                break;
            case 'f':
                file_io_flag = true;
                break;
```

```
                case 'd':
                    seconds = atoi(optarg);
                    break;
                case 'w':
                    delay = atoi(optarg);
                    break;
                default:
                    std::cout << "Unknown command detected, program terminated\n";
                    std::cout << "Usage: ./dispatch_test \n";
                    std::cout << "    [-i irq_number]    irq line number to test\n";
                    std::cout << "    [-e]    attach to event instead of function\n";
                    std::cout << "    [-f]    write time results to file\n";
                    std::cout << "    [-d seconds]    test duration in seconds\n";
                    std::cout << "    [-w seconds]    test start delay in seconds\n";
                    break;
            }
        }
```

Перевірка виконана за допомогою методу покриття умов за тестування наступних комбінацій параметрів:

1. -i0 -d2 -w2 -e -f
2. -f2 -w3 -e1
3. -u -e -i9 -d11

Результати наведемо у вигляді таблиці. Стовпці цієї таблиці позначають оператор case, якого досягла програма при обробці отриманих значень. Строки позначають номер тесту з переліку нижче. Покриття умови позначене знаком «+», відсутність покриття — знаком «-».

Таблиця 3.2 — Тестування коректності зчитування параметрів користувача методом покриття операторів

	case 'i'	**case** 'e'	**case** 'f'	**case** 'd'	**case** 'w'	default
1	+	+	+	+	+	-
2	-	+	+	-	+	-
3	-	-	-	-	-	+

3.4.2 Перевірка коректності переключення контекстів потоків користувача та ядра

Під час тестової роботи з модулем планування обробки переривання було помічено, що програмою не повертались ніякі результати експериментальних замірів та статистичних розрахунків. Методом припущення про помилку було висунуто гіпотезу про неправильне переключення між контекстами потоків ядра та користувача, що заважало коректному відображенню результатів для розрахунку та запису у відповідні структури даних. Дослідження вихідного коду підтвердило висунуту гіпотезу — функцією потоку ядра, що відповідає за обробку переривань, замість події повертався вказівник NULL, хоча для правильного переключення контексту важливо використовувати дану структуру, атрибути якої були визначені макросом SIGEV_INTR_INIT.

Код потоку обробки переривання до налагодження:

```
const struct sigevent *handler(void *area, int id)
{
        cycle2 = ClockCycles( ); //фіксація часу у циклах системного таймеру
        return(NULL);
}
```

Код потоку обробки переривання після налагодження:

```
const struct sigevent *handler(void *area, int id)
{
        cycle2 = ClockCycles( ); //фіксація часу у циклах системного таймеру
        return(&event);
}
```

Висновки за розділом 3

У даному розділі були розглянуті усі програмні аспекти проектування інструментальних засобів для дослідження затримок обробки переривань у ОСРЧ QNX. Разом з цим на даному етапі було почато розробку програмного забезпечення автоматизації досліджень, результати яких будуть надані у наступному розділі. Як було визначено у попередньому розділі, бібліотека системних викликів QNX міс-

тить достатньо функцій для розробки програм обробки переривань та моделювання випадків затримки цієї обробки. Розглянуті алгоритми моделювання були використані для написання програмного коду моделювання трьох основних випадків затримки обробки переривання з використанням системних викликів QNX. Тестування та налагодження розроблених програмних модулів дозволило віднайти та виправити помилки, що спричиняли некоректну роботу програм.

4 ДОСЛІДЖЕННЯ ЗАТРИМКИ ОБРОБКИ ПЕРЕРИВАНЬ В ОСРЧ QNX 6

4.1 Вихідні умови експериментів

Експерименти з дослідження часових характеристик затримки обробки переривань повинні проводитись для їх аналізу перед отриманням статистичних даних, що потім будуть використовуватися для формування результативних висновків.

Для поточного експерименту було виділено умови проведення на основі доступних для використання ресурсів, тому існує вірогідність того, що для більш ранніх або пізніх версій операційної системи реального часу QNX результати проведення аналогічних експериментів можуть суттєво відрізнятися. У даному розділі будуть розглянуті умови проведення експерименту

4.1.1 Опис програмно-апаратного середовища для проведення експерименту

Дослідження затримок обробок переривань проводиться на двох ЕОМ — фізичній та віртуальній. Фізична машина представляє собою персональний IBM-сумісний комп‘ютер зі встановленою ОСРЧ QNX 6.3.2. Віртуальна машина реалізується за допомогою гіпервізора VMware Workstation Player 15, що встановлений на іншому персональному IBM-сумісному комп‘ютері з операційною системою Microsoft Windows 7. У таблиці 4.1 наведений перелік деяких системних характеристик для обох машин та їх значення.

Таблиця 4.1 — Системні характеристики використаних у дослідженнях ЕОМ

	Віртуальна машина	«Чиста» QNX
Модель процесору	Intel Core i3-2350M	Intel Pentium D
Тактова частота процесору, ГГц	2,30	3,40
Кількість ядер	1	1
Об‘єм оперативної пам‘яті, МБ	1023	1024
Підключені PCI-пристрої	Вбудовані клавіатура та сенсорна миша (тачпад), DVD-дисковод, жорсткий диск об‘ємом 450 Гб	PS/2-клавіатура, USB-миша, жорсткий диск об‘ємом 256 Гб

4.1.2 Опис використаних системних викликів ОСРЧ QNX 6

Системні виклики та функції, визначені бібліотекою QNX Neutrino у файлі «sys/neutrino.h», що були використані при написанні потоків обробки переривань, були розглянуті у попередніх розділах.

4.1.3 Порівняння результатів експерименту для фізичної та віртуальної машин

Відповідно до табл. 4.1, машини, що використовуються для проведення досліджень, мають різні характеристики центрального процесору, але налаштування віртуальної машини за допомогою відповідних опцій гіпервізора для деяких характеристик, як-то оперативна пам‘ять та кількість виділених ядер процесору, були вручну встановлені такими, що близькі до значень або однакові з фізичною машиною.

Враховуючи наявність відмінностей у конфігурації обох машин, за результатами дослідження затримки планування слід визначити, чи мають вплив на результати як апаратні характеристики, так і використання віртуальних машин при проведенні експериментів. Відповіді на дані питання при виконанні замірів експериментальних значень та їх подальшому статистичному аналізі допоможуть визначити, яку з двох машин доцільніше використовувати для проведення подальших досліджень та формування остаточних статистичних даних щодо часових характеристик затримки обробки переривань.

4.1.4 Перелік ліній переривань для тестування

Таблиця 4.2, представлена нижче, надає перелік усіх номерів ліній переривань, що плануються розглядатись у рамках проведення даного дослідження. Дані надаються для використаних віртуальної та фізичної машин зі встановленою «чистою» ОСРЧ QNX.

Таблиця 4.2 — Перелік досліджуваних ліній переривань

Номер лінії		Номер пріоритету		Джерело перериванння
Для віртуальної машини	Для «чистої» QNX	Для віртуальної машини	Для «чистої» QNX	
0		2		системний таймер
1		3		клавіатура
12	5	8	14	комп‘ютерна миша
14	5	10	14	1-й контролер жорсткого диску HDD

Конфігурація пристроїв та відповідних їм ліній переривання є стандартною та не була змінена під час проведення експериментів. Як можна побачити з таблиці вище, деякі пристрої, під'єднанні до фізичної ЕОМ, мають спільну лінію переривань, що необхідно буде враховувати під час проведення тестів з часових замірів обробки переривань цих пристроїв.

Оскільки для проведення даного дослідження використовуються дві різні за своїми характеристиками ЕОМ, за результатами проведення дослідження буде визначено, чи мають вплив ці характеристики на отримані статистичні дані.

4.2 Проведення експериментів

Для кожного з трьох випадків затримки обробки переривань у ОСРЧ QNX, що були змодельовані за допомогою програмних засобів операційної системи та описані у попередньому розділі, був проведений ряд тестів на отримання часових характеристик затримки обробки переривань та виконання статистичної обробки отриманих результатів.

4.2.1 Заміри часу затримки планування обробки переривань

Початкові тести затримки планування при стандартній завантаженості системи проводяться за наявності робочих процесів, що вказані у переліку на рис. 4.1 нижче (дані отримані з графічного додатку System Information, встановленого за умовчанням у ОСРЧ QNX).

Name	Pid	Code	Data	Stack	Vstack	CPU
procnto	1	0	0	0	0	301682
pwm	376860	80K	196K	12K	516K	24
shelf	421920	88K	680K	24K	648K	67
qde	512037	16K	132K	12K	516K	4
pvm	512038	29M	82M	160K	7074K	19720
pterm	507930	48K	208K	12K	516K	5
sh	507933	144K	120K	8K	516K	2
qconn	507940	104K	140K	20K	912K	0
psin	507917	28K	456K	12K	516K	139
bkgdmgr	458785	12K	6358K	12K	516K	396
wmswitch	458786	8K	132K	12K	516K	2
saver	458787	16K	232K	12K	516K	22
Photon	167963	68K	244K	8K	516K	220
phfont	204812	4K	2270K	24K	1033K	212
fontsleuth	217105	40K	120K	24K	1429K	4
devi-hid	266271	72K	160K	28K	668K	44
io-graphics	245790	108K	4820K	32K	1165K	11786
io-net	90127	64K	280K	80K	908K	8
tinit	2	8K	84K	8K	516K	0
login	278553	16K	84K	8K	516K	0
login	122904	16K	84K	8K	516K	1
login	122903	16K	84K	8K	516K	2
login	122902	16K	84K	8K	516K	2
dumper	106517	20K	84K	8K	516K	1
random	94228	20K	324K	20K	780K	2
devc-pty	90131	44K	216K	8K	516K	1
mqueue	20491	12K	84K	8K	516K	1
pipe	16394	16K	116K	40K	564K	5
umass-enum	8201	24K	172K	16K	536K	3
devb-eide	8200	68K	164M	112K	660K	2524
devc-con-hid	4103	68K	120K	12K	536K	7
io-hid	4102	28K	128K	32K	600K	171
io-usb	4101	68K	420K	44K	696K	592
slogger	4100	8K	152K	8K	516K	1
pci-bios	4099	48K	88K	8K	516K	29

Рисунок 4.1 — Перелік процесів при стандартній завантаженості системи

Статистичні дані за отриманими експериментальними значеннями були отримані у табличному процесорі Microsoft Excel. Нижче представлена таблиця отриманих показників.

Таблиця 4.3 — Статистичні показники затримки диспетчеризації обробки переривань для переривань клавіатури, мс

	Віртуальна машина	«Чиста» QNX	Різниця, %
Мат. очікування	33,3319362	37,4127902	10,91%
Дисперсія	3,564942092	0,679603401	-80,94%
Станд. відх.	1,888105424	0,824380617	-56,34%

У даному випадку можливо побачити, що показник математичного очікування для фізичної ЕОМ є більшим за значення для віртуальної машини майже на 11%, однак стандартне відхилення є меншим майже вдвічі за віртуальну машину (56,34%). Це можливо пояснити конфігурацією обох ЕОМ — процесор віртуальної машини має більшу актову частоту, але наявність віртуалізації вносить деякі погрішності у часові показники роботи. Виконаємо візуальний аналіз часових характеристик на предмет виявлення погрішностей, спричинених віртуалізацією. Для цього на базі отриманих часових замірів у табличному процесорі Microsoft Excel було побудовано два графіки зміни часових замірів у часі для перших та останніх 50-ти значень замірів.

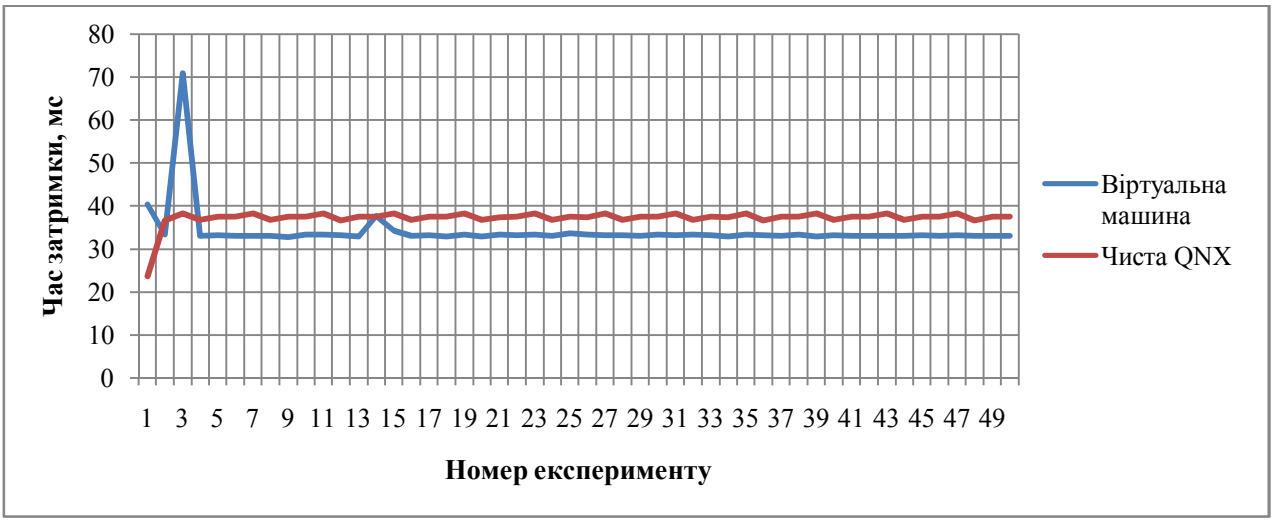

Рисунок 4.2 — Графік отриманих значень часу затримки для переривань клавіатури (перші 50 значень), мс

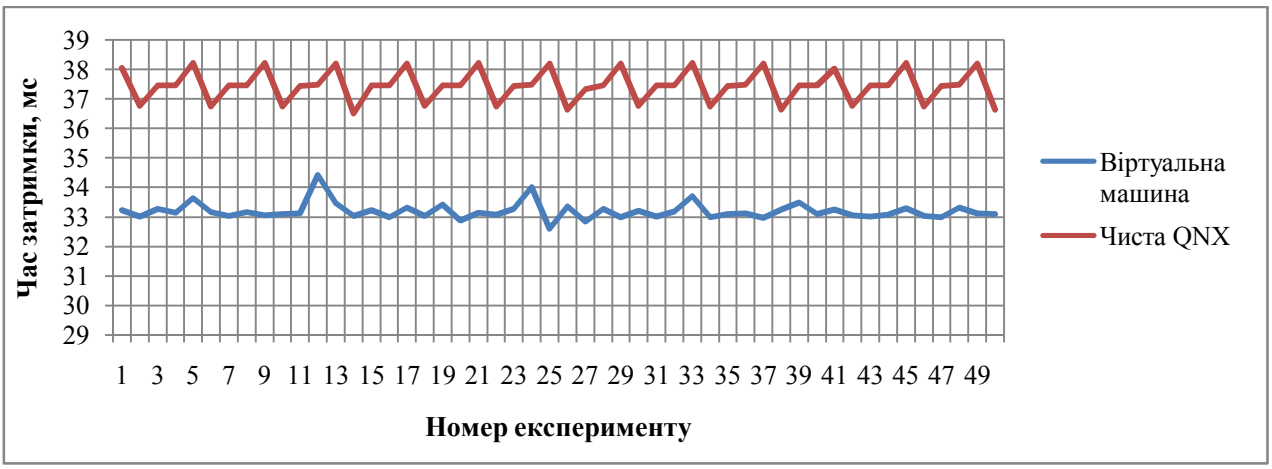

Рисунок 4.3 — Графіки отриманих значень часу затримки для переривань клавіатури (останні 50 значень), мс

На другому графіку явно спостерігається монотонність зміни часових значень затримки обробки для фізичної ЕОМ, у той час, як значення для віртуальної машини нерівномірно коливаються до появи аномально високих значень, як показано на першому графіку.

Таблиця 4.4 — Статистичні показники затримки диспетчеризації обробки переривань для переривань миші, мс

	Віртуальна машина	«Чиста» QNX	Різниця, %
Мат. очікування	941,9874454	22,55718885	-97,61%
Дисперсія	865352,4049	18557,24911	-97,86%
Станд. відх.	930,2431966	136,2249944	-85,36%

Отримані часові показники для віртуальної машини перевищують значення для фізичної ЕОМ на більш, ніж 85%. Розглянемо результати часових замірів у графічному вигляді.

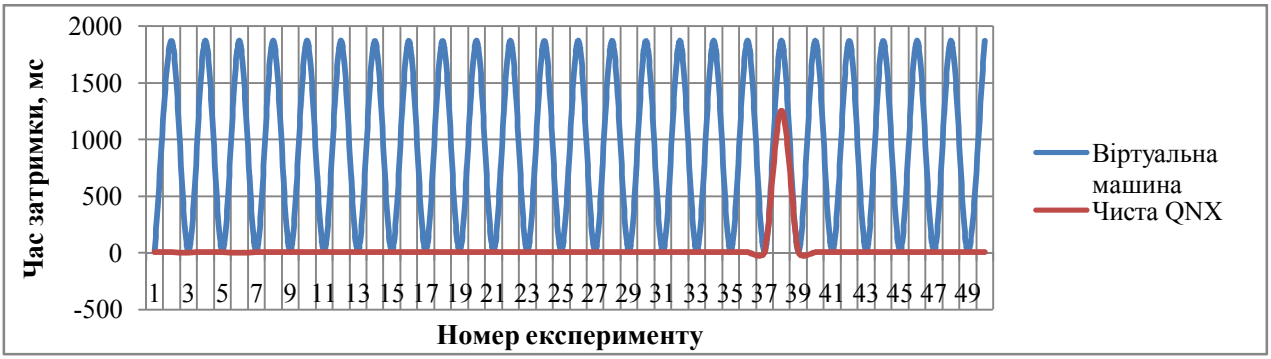

Рисунок 4.4 — Графік отриманих часових даних для переривань миші, мс

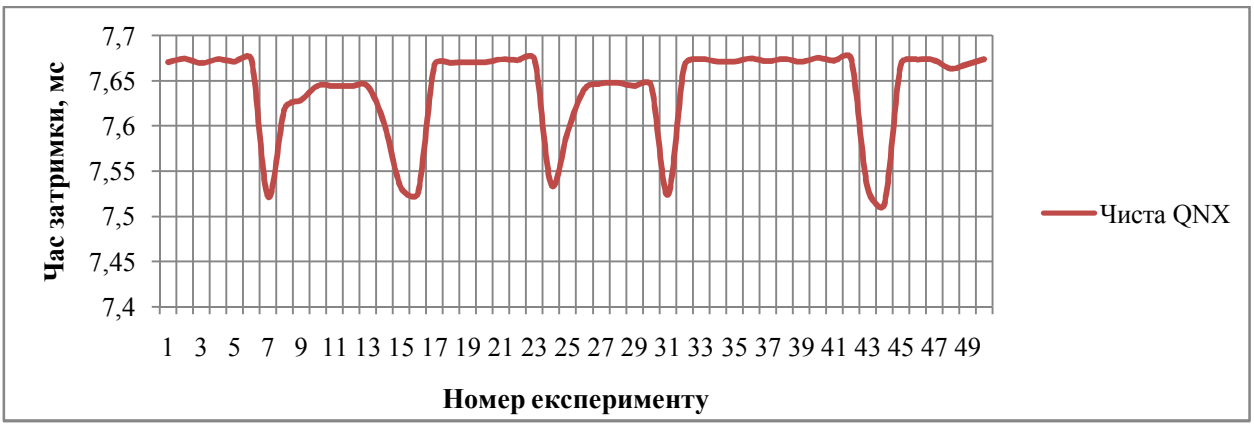

Рисунок 4.5 — Збільшений частковий графік отриманих часових даних для переривань миші для «чистої» QNX, мс

У даному випадку спостерігаються великі коливання значень для віртуальної машини, що відбуваються з певною періодичністю, у той час, як для фізичної машини спостерігається лінійність графіку з місцевим виникненням аномальних коливань, що спричинені наявністю інших переривань, що розділяють з мишею одну й ту саму лінію переривань. У порівнянні з віртуальною машиною значення для «чистої» QNX знаходяться у меншому діапазоні, що свідчить про наявність менш тривалої затримки.

Таблиця 4.5 — статистичні показники затримки диспетчеризації обробки переривань для переривань системного таймеру, мс

	Віртуальна машина	«Чиста» QNX	Різниця, %
Мат. очікування	8,377372841	3,485552305	-58,37%
Дисперсія	13961,09372	3158,538811	-77,38%
Станд. відх.	118,1570722	56,20087909	-52,44%

Знову спостерігається значне зменшення статистичних показників для фізичної машини у порівнянні з віртуальною на більш, ніж 50%. Крім того, отримані значення є найменшими з розглянутих у рамках первинних досліджень. Візуалізація отриманих часових показників зображена на графіках нижче.

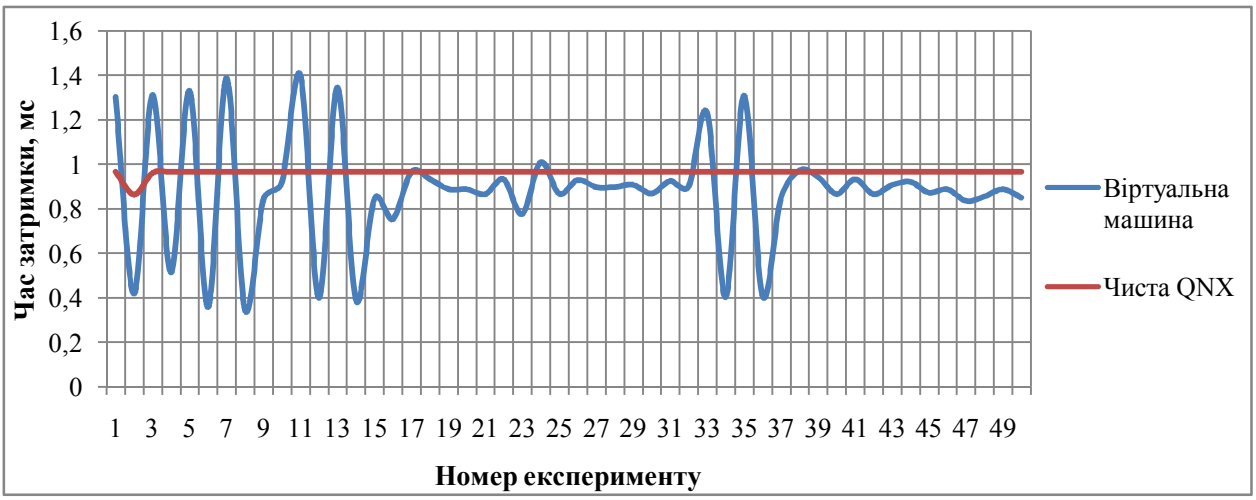

Рисунок 4.6 — Графік отриманих значень планування обробки переривань для системного таймеру

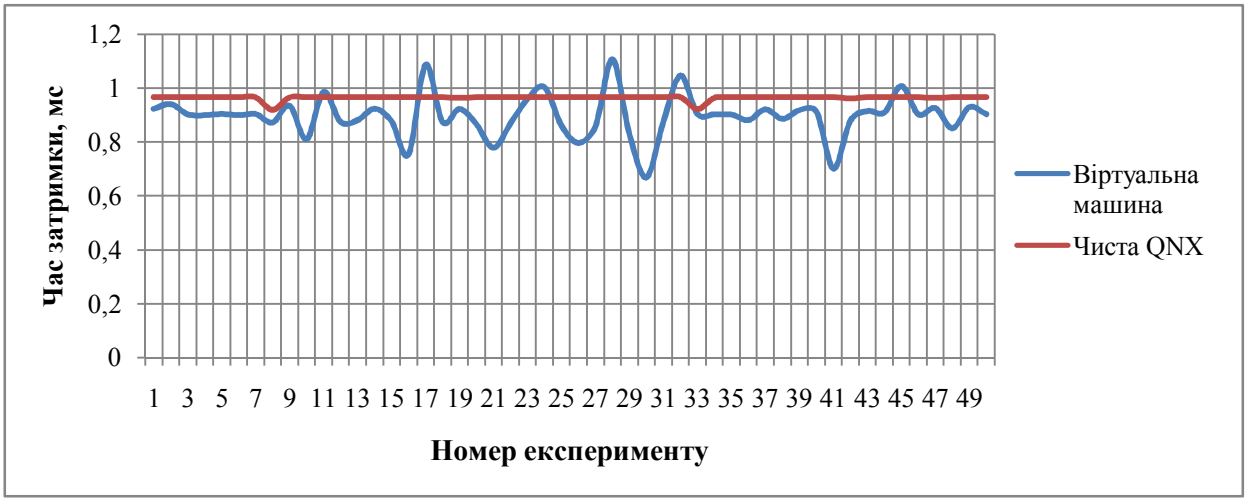

Рисунок 4.7 — Продовження графіку отриманих значень планування обробки переривань для системного таймеру

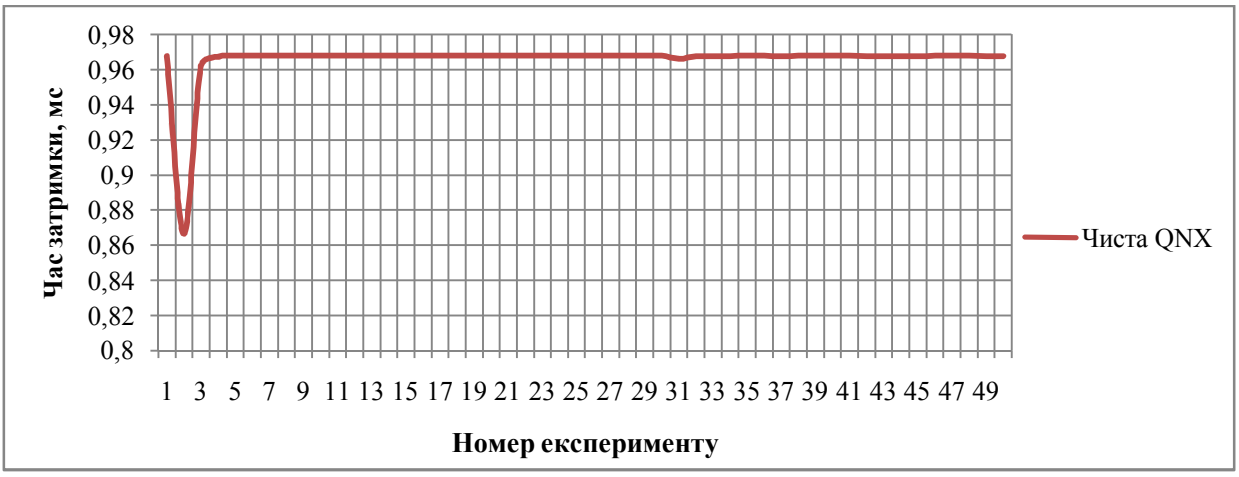

Рисунок 4.8 — Збільшений графік значень планування обробки переривань для фізичної машини для системного таймеру для фізичної ЕОМ

Порівнюючи отримані графіки, можна побачити, що значення затримки обробки переривань системного таймеру без урахування значних відхилень тримається майже на константному значенні для фізичної машини, у той час, як на графіку віртуальної машини помітні значні та нерегулярні коливання, діапазон яких зменшується у часі.

Виконаємо дослідження часу затримки планування при значному навантаженні системи фізичної ЕОМ та порівняємо отримані результати з простим плануванням. Для цього була створена примітивна програма, що генерує 100 процесів та запускається окремо під час роботи процесу з потоком обробки переривання. У якості досліджуваних ліній переривань було узято переривання системного таймеру та першого контролеру жорсткого диску HDD. Тут і далі усі заміри часу виконуються у середовищі фізичної машини зі встановленою QNX у мікросекундах (мкс).

Таблиця 4.6 — Статистичні дані щодо затримки планування при високому рівні навантаження системи, мкс

	Системний таймер		1-й контролер HDD	
Прив‘язка переривання	До функції	До події	До функції	До події
Мат. очікування	988.526930025	994.897044762	698.576388812	772.793787182
Дисперсія	339.905224351	404.887688996	1012992.4132	3796352.56877
Станд. відхилення	18.4365187698	20.1218212147	1006.47524222	1948.42309799
Мінімум	410.161734544	365.099033109	65.180779373	40.3656607091
Максимум	992.136536771	1028.07910929	6317.86346323	60629.4664518

Порівняння даних для різної ступені навантаження для переривань, прив‘язаних до функції, представлені у таблицях нижче.

Таблиця 4.7 — Порівняння часу затримки планування системного таймеру, мкс

	Час без значного навантаження, мкс	Час зі значним навантаженням, мкс	Різниця, %
Мат. очікування	988,7338584	988,52693	-0,02%
Дисперсія	320,1579199	339,9052244	5,81%
Станд. відх.	17,89295727	18,43651877	2,95%

Таблиця 4.8 — Порівняння часу затримки планування контролеру HDD, мкс

	Час без значого навантаження, мкс	Час зі значним навантаженням, мкс	Різниця, %
Мат. очікування	711,6080404	698,5763888	-1,83%
Дисперсія	997322,7877	1012992,413	1,55%
Станд. відх.	998,6604967	1006,475242	0,78%

При навантаженні системи значення математичного очікування зменшується, однак стандартне відхилення часових значень є більшим за значення для стандартного навантаження. Втім, різниця у часі для обох показників є незначною, що дає підстави її нехтуванням при обчисленні статистичних даних за різних умов системного середовища.

4.2.2 Заміри часу на затримку потоку відкладеного обробника переривання

Для проведення даного дослідження використовується екземпляр структури події sigevent, до якого прив‘язується функція обробки, що й є відкладеним обробником. Ця функція запускається у відповідь на подію, що повертається функцією ядра, прив‘язаної до відповідного сигналу переривання. Заміри часу затримки відкладених обробників відбуваються без урахування часу на переключення ядром контексту з робочого потоку на потік прив‘язаної функції.

Таблиця 4.9 — статистичні показники затримки відкладеної обробки переривань, мкс

	Системний таймер	Клавіатура	1-й контролер HDD
Мат. очікув.	33,8903432	49,93895	39,89756
Дисперсія	14,5015121	46,82224	30,74265
Станд. відх.	3,8080851	6,842678	5,544606
Мінімум	23,6894	35,6636	25,4776
Максимум	79,8497	79,2669	71,9449

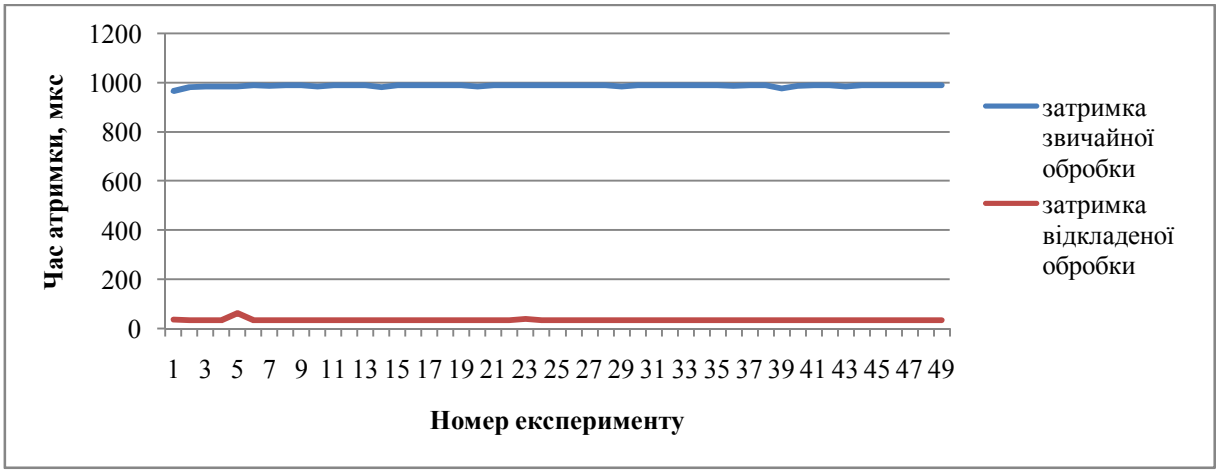

Рисунок 4.9 — Графік часових значень затримки відкладеної обробки переривань системного таймеру у порівнянні зі звичайною обробкою

Отримані характеристики є значно меншими у порівнянні з отриманими даними для планування, однак їх наявність не означає відсутності затримки на переключення контексту на потік планування відкладених переривань. Крім того, для різних ліній переривань спостерігаються деякі відмінності — найменші дані у таблиці вище належать часовим показникам, отриманим з досліджень переривань системного таймеру. Відкладені функції обробки переривання також були окремо розглянуті під час дослідження третього випадку виникнення їх затримки, що представлений у підрозділі нижче.

4.2.3 Заміри часу на затримку потоку обробки переривання нижчого пріоритету

Для проведення даного дослідження програмою, що моделює ситуацію затримки обробки переривання за пріоритетністю, планується до роботи потік обробки переривань системного таймеру під час роботи потоку, що обробляє переривання на іншій лінії з нижчим пріоритетом (у випадку даного дослідження — переривання контролера жорсткого диску).

Таблиця 4.10 — Статистичні показники обробки переривань, мкс

	Час самостійної роботи	Час роботи паралельно потоку високого пріор.	Потік високого пріоритету
Мат. очікув.	0,13462072	0,132449767	0,063911991
Дисперсія	0,00012633	0,027234928	0,000286712
Станд. відх.	0,01123983	0,165030082	0,016932569
Мінімум	0,07971864	0,05978898	0,03985932
Максимум	0,25908558	6,053634232	0,224208675

Таблиця 4.11 — Порівняння отриманих статистичних значень для потоку низького пріоритету

	Час самостійної роботи, мкс	Час роботи паралельно потоку високого пріор., мкс	Різниця, %
Мат. очікув.	0,13462072	0,132449767	-1,61%
Дисперсія	0,00012633	0,027234928	99,54%
Станд. відх.	0,01123983	0,165030082	93,19%
Мінімум	0,07971864	0,05978898	-25,00%
Максимум	0,25908558	6,053634232	95,72%

Таблиця 4.12 — Порівняння отриманих статистичних значень для паралельно виконуваних потоків обробки переривань нижчого та вищого пріоритетів.

	Час роботи паралельно потоку високого пріор., мкс	Потік високого пріоритету, мкс	Різниця, %
Мат. очікув.	0,132449767	0,063911991	-51,75%
Дисперсія	0,027234928	0,000286712	-98,95%
Станд. відх.	0,165030082	0,016932569	-89,74%
Мінімум	0,05978898	0,03985932	-33,33%
Максимум	6,053634232	0,224208675	-96,30%

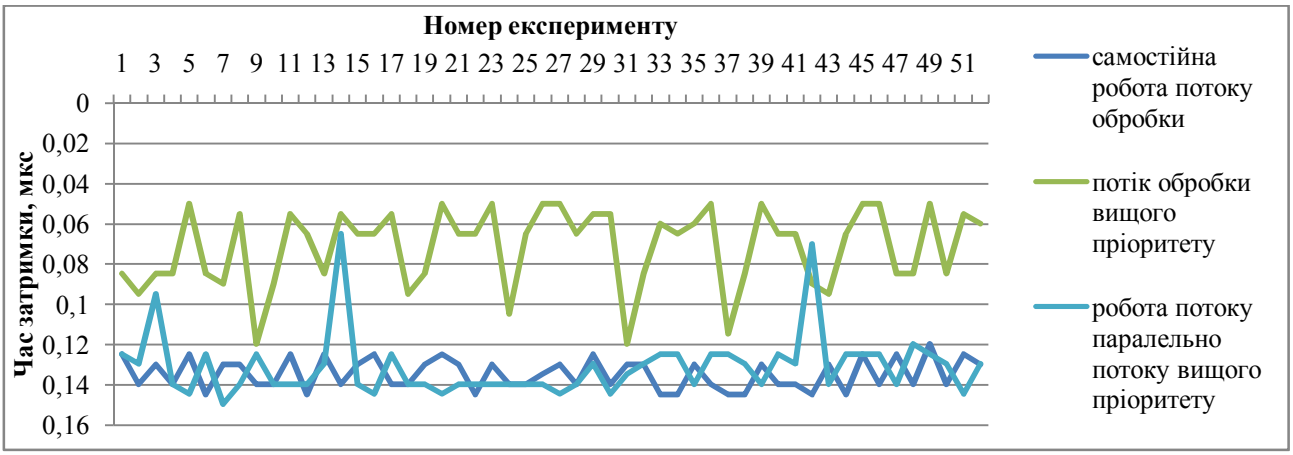

Рисунок 4.10 — Графік часових замірів роботи потоків за пріоритетом

Виконаємо аналогічне тестування для відкладеної обробки потоку з нижчим пріоритетом.

Таблиця 4.13 — Статистичні показники затримки обробки переривань з вищим та нижчим пріоритетами, мкс

	Час самостійної роботи відкладеного обробника	Час роботи паралельно потоку високого пріор.	Потік високого пріоритету
1	2	3	4
Мат. очікув.	0,131813325	0,132122454	0,083543887
Дисперсія	0,001002192	0,000245246	0,000243322

Продовження таблиці 4.13

1	2	3	4
Станд. відх.	0,031657413	0,01566032	0,015598785
Мінімум	0,074714328	0,039847641	0,039847641
Максимум	1,050981541	0,219162028	0,204219162

Таблиця 4.14 — Порівняння отриманих статистичних значень для потоку відкладеної обробки переривання низького пріоритету

	Час самостійної роботи, мкс	Час роботи паралельно потоку високого пріор., мкс	Різниця, %
Мат. очікув.	0,131813325	0,132122454	0,23%
Дисперсія	0,001002192	0,000245246	-75,53%
Станд. відх.	0,031657413	0,01566032	-50,53%
Мінімум	0,074714328	0,039847641	-46,67%
Максимум	1,050981541	0,219162028	-79,15%

Таблиця 4.15 — Порівняння отриманих статистичних значень для паралельно виконуваних потоків обробки переривань нижчого та вищого пріоритетів

	Час роботи паралельно потоку високого пріор., мкс	Потік високого пріоритету, мкс	Різниця, %
Мат. очікув.	0,132122454	0,083543887	-36,46%
Дисперсія	0,000245246	0,000243322	-12,69%
Станд. відх.	0,01566032	0,015598785	-6,56%
Мінімум	0,039847641	0,039847641	0,00%
Максимум	0,219162028	0,204219162	-36,36%

У рамках даного експерименту також було досліджено часові дані з затримкою початку роботи потоку обробки на 8 мс. Затримка була реалізована системним викликом nap() відкладення роботи процесу.

Таблиця 4.16 — Статистичні показники затримки обробки переривань нижчого пріоритету з затримкою роботи обробки у 8 мс, мкс

	Клавіатура	1-й контролер HDD
Мат. очікув.	9728.91821224	2222.07784701
Дисперсія	125358.170088	3752575.15455
Станд. відх.	354.059557261	0,01566032
Мінімум	9133.43262102	1937.15646104
Максимум	10515.6641954	10316.7841256

З отриманих результатів видно, що затримка роботи потоку обробки переривань на 8 мс значним чином вплинула на час роботи потоку та на вищезазначені статистичні показники.

4.2.4 Дослідження впливу затримки обробки переривань на значення WCET

Наостанок в ході проведення дослідження було проведено експериментальні заміри метрики WCET для прикладів коду різного призначення з використанням обробки переривань. Метою проведення даних експериментів є визначення відсоткової частки від WCET, що припадає на безпосередньо затримку обробки переривань, а також істотності впливу затримки на значення WCET. Отримані результати допоможуть сформувати висновки про необхідність приділення уваги затримці переривань для задач різного об‘єму.

Для виконання даного експерименту було розроблено дві прості програми, що використовують обробку переривань різним чином. Нижче наведені значення математичного очікування часу виконання програм та математичні очікування відповідних їм затримок обробки переривань разом з описом цих програм. Першою у даному експерименті перевіряється елементарна програма обробки переривань, що виконується в одному й єдиному робочому потоці програмного процесу.

Таблиця 4.17 — Часові дані для елементарної програми обробки переривань контролеру жорсткого диску, мкс

Математичне очікування для програми	Математичне очікування для затримки переривання	Відсоток часу затримки відносно виконання програми, %
9087,558514	9073,330032	99,84
7674,880935	7660,340106	99,81
389,7780627	375,8719431	96,43
564,3317409	553,0658529	98,00
602,5811928	592,7780422	98,37
396,8152755	386,9763013	97,52
374,1097626	362,8081008	96,98
365,1453517	351,4283646	96,24
353,323476	343,5035844	97,22
375,9971512	366,18	97,39

Таблиця 4.18 — Часові дані для елементарної програми обробки переривань клавіатури, мкс

Математичне очікування для програми	Математичне очікування для затримки переривання	Відсоток часу затримки відносно виконання програми, %
1	2	3
40222,31912	40203,42785	99,95
18894,23093	18875,58589	99,90
18893,22642	18874,66548	99,90
18894,57751	18876,22264	99,90
18893,40773	18874,45901	99,90
18892,01141	18873,16089	99,90
18893,69781	18875,14803	99,90

Продовження таблиці 4.18

1	2	3
18893,23021	18874,85726	99,90
18894,87431	18876,20116	99,90
18893,33479	18874,53788	99,90

Таблиця 4.19 — Часові дані для елементарної програми обробки переривань системого таймеру, мкс

Математичне очікування для програми	Математичне очікування для затримки переривання	Відсоток часу затримки відносно виконання програми, %
508,0270135	498,2383118	98,07
498,7219549	489,5690094	98,16
499,0037104	490,7312075	98,34
499,3198447	490,7057972	98,27
498,8547362	485,7799326	97,38
499,2955305	489,7135991	98,08
498,62983	486,2423007	97,52
498,7877227	485,8337925	97,40
499,0158675	486,2534613	97,44
498,9106887	490,3105422	98,28

Як можна побачити з таблиць вище, час на обробку переривань для усіх розглянутих ліній переривань у відсотках становить майже увесь час, витрачений на роботу даної функції, з незначною різницею у відсотковій частці (не більш, ніж 2,5%) часу затримки між розглянутими лініями переривань.

Розглянемо характеристики програми комунікації потоків клієнта та сервера, у якій використовується обробка переривань з прив‘язкою до функції. Потоки у програмі обмінюються між собою повідомленнями всього один раз за умови, якщо потоком клієнта було отримано сигнал про виникнення переривання на тій чи іншій

лінії. У результаті отримання повідомлення від клієнта сервер циклічно виконує деяку операцію з перетворення даних з одного типу у інший та надсилає повідомлення у відповідь. Далі наводяться таблиці з аналогічними попереднім замірами та розрахунками математичної статистики для усього процесу та затримки обробки переривання окремо.

Таблиця 4.20 — Часові дані для програми комунікації потоків з використанням обробки переривань клавіатури, мкс

Математичне очікування для програми	Математичне очікування для затримки переривання	Відсоток часу затримки відносно виконання програми, %
59800,34615	54293,95453	90,79
38558,69569	32311,15353	83,80
38171,87563	32088,53276	84,06
37798,76913	32184,67005	85,15
37798,91472	32114,80996	84,96
37799,19702	32247,12681	85,31
37797,28875	32289,84609	85,43
38562,37181	32685,34805	84,76
38167,82468	32352,63398	84,76
38177,36331	32597,55042	85,38

Таблиця 4.21 — Часові дані для програми комунікації потоків з використанням обробки переривань контролеру HDD, мкс

Математичне очікування для програми	Математичне очікування для затримки переривання	Відсоток часу затримки відносно виконання програми, %
1	2	3
73288,05458	314,6355715	0,43
70103,56012	328,7051641	0,47

Продовження таблиці 4.21

1	2	3
72427,9751	405,3544373	0,56
73088,01669	406,9935522	0,56
72333,39232	302,4245692	0,42
73773,25162	299,1678136	0,41
73845,78098	212,8159848	0,29
71220,77974	280,6390592	0,39
73906,22171	387,7795164	0,52
73468,25413	327,3365445	0,45

Таблиця 4.22 — Часові дані для програми комунікації потоків з використанням обробки переривань системного таймеру, мкс

Математичне очікування для програми	Математичне очікування для затримки переривання	Відсоток часу затримки відносно виконання програми, %
6404,583587	277,5640123	4,33
3798,288426	350,1468376	9,22
7325,910445	391,7034672	5,35
3809,764024	373,2860698	9,80
6219,184487	372,804619	5,99
4864,749097	332,4553283	6,83
3821,161298	325,0355158	8,51
7305,940179	301,762755	4,13
3768,403901	315,1941002	8,36
7308,983438	361,6855129	4,95

Отримані значення є помітно меншими за значення для попередньої програми через більший об‘єм операцій, виконуваних програмою потоків клієнту та серве-

ру, що досліджується, однак переривання клавіатури все ще становлять дуже велику частку загального часу виконання програми.

4.3 Аналіз результатів проведення експериментів

4.3.1 Вплив віртуалізації та системних характеристик на часові показники

У результаті початкових експериментів було відмічено більш швидку роботу системного ядра та наявність менших за значенням та кількістю відхилень значень для фізичної машини у порівнянні з віртуальною, незважаючи на більшу тактову частоту процесора віртуальної машини. Менші показники для віртуальної машини пояснюються наявністю погрішностей, внесених віртуалізацією середовища, і саме тому за отриманими результатами початкових експериментів у дослідженнях затримок відкладених обробок переривань та обробок переривань нижчого пріоритету у подальших експериментах було надано перевагу проведенням досліджень у «чистому» середовищі ОСРЧ QNX.

4.3.2 Вплив завантаженості системи на значення затримки переривань

За результатами тестування було визначено, що завантаженість системи має незначний вплив на часові показники — для математичного очікування спостерігається зменшення значення у разі збільшення навантаження, що не перевищує 2%, для стандартного відхилення спостерігається збільшення числового значення у разі навантаження до +3%. Такими даними щодо різниці можливо знехтувати для переривань, наприклад, системного таймеру. Результуючі показники та їх незначна відмінність пояснюється саме особливостями реалізації архітектури ОСРЧ QNX, оскільки розв‘язок задач реального часу потребує наявності програмного середовища, стійкого до навантаження.

4.3.3 Аналіз впливу часових характеристик затримки переривань на метрику найгіршого часу виконання програм WCET

Отримані результати для двох випадків використання показали, що для окремих підпрограм або методів, що відповідають тільки за елементарну обробку переривань, затримка потоку обробки займає майже увесь час, витрачений програмою на виконання. Для більш складних програм частка часу, витраченого на цю за-

тримку, від загального часу варіюється в залежності від переривання. Для таких переривань, як переривання системного таймеру та контролеру жорсткого диску, цей час є дуже малим, у той час, як час, витрачений на диспетчеризацію потоку обробки переривань від клавіатури становить більше, ніж 85% загального часу виконання. Отже, за цими спостереженнями можемо зробити наступні висновки про вплив затримки переривань на метрику WCET:

- найбільший вплив затримки на WCET присутній для програм, що призначені виключно для обробки переривань;
- переривання периферійних пристроїв, як-от клавіатури, можуть мати значний вплив на WCET для складних багатофункціональних програм, у той час, як вплив переривань системного середовища значно зменшується зі зростанням об‘єму програми.

Висновки за розділом 4

Дослідження, описані у даному розділі, дали змогу більш детально вивчити особливості обробки переривань у середовищі операційної системи реального часу QNX у плані часових характеристик та статистичних даних на їх основі. Затримка обробки переривань є непостійною у часі та залежить від безлічі факторів, зокрема, ліній переривання та пристроїв, що під‘єднані до них, процесу, що використовує обробку сигналів від ліній переривань тощо.

5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Робоче місце, або робоча станція, описується у Вимогах щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями НПАОП 0.00-7.15-18 [22] як сукупність устаткування, що включає екранний пристрій, що може доповнюватися периферійними пристроями вводу-виводу інформації, підключення до мережі та/або іншого призначення, а також робочим кріслом, робочим столом або робочою поверхнею «розумного» столу. Дослідження затримки обробки переривань, включаючи використання розробленого у результаті проведення даної дослідницької роботи програмного комплексу, потребує постійної взаємодії з робочою станцією, та особливо з ЕОМ та складовими апаратними компонентами, що тестуються. У даному розділі наведені вимоги до дотримання безпеки на робочому місці під час проведення аналогічного дослідження, а також надана інформація щодо небезпечних виробничих факторів та дій при можливих аварійних ситуаціях на робочому місці.

5.1 Вимоги безпеки праці під час виконання робіт з дослідження затримок обробки переривань

Дослідження затримок обробки переривань передбачає наявності комунікацій робочого місця з пристроями, обробка переривання яких будуть досліджуватись на предмет часових значень. З поставленої дослідником задачі випливають відповідні вимоги організації робочого місця та простору його розташування.

5.1.1 Характеристики обладнання для дослідження

У даному підрозділі будуть розглянуті вимоги до організації робочої ЕОМ (екранного пристрою) та робочого місця оператора.

Для проведення даного дослідження та розробки відповідного програмного забезпечення використовується персональний комп'ютер зі стандартною конфігурацією, тобто, складається з наступних компонентів:

- системний блок, що містить усі апаратні компоненти ЕОМ відповідно до поставлених у технічному завданні системних вимог, такі, як центральний процесор, оперативна пам'ять, жорсткий HDD-накопичувач та ін. З'єднання компонентів за допомогою шин та адап-

терів, а також підключення усього блоку до електромережі не повинно бути загрозою ураження електричним струмом або замикання електромережі — усі пристрої повинні бути справними та не мати будь-яких механічних пошкоджень. Крім того, корпус та внутрішній склад системного блоку повинен мати належну вентиляцію та систему охолодження для уникнення перегріву електронних компонентів. Сам системний блок повинен бути розміщений у відкритому місці якомога далі від будь-яких джерел тепла та вологи;

- VGA-сумісний монітор, екран якого не повинен перенавантажувати зір оператора, зокрема, він не повинен відблискувати або відбивати світло, щоб не викликати дискомфорту у оператора під час роботи. Зменшення навантаження на зір також можливо за допомогою використання оператором спеціальних окулярів під час роботи;
- клавіатура та миша, положення яких на робочому місці не повинно завдавати дискомфорту операторові у їх використанні та переміщенні робочим місцем.

Усі пристрої та компоненти обладнання робочого місця повинні відповідно до НПАОП 0.00-7.15-18 [22] обиратись такими, що не створюють зайвого шуму та не виділяють надлишкового тепла. Взагалі, рівні шуму та температура робочого простору мають відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [23]. Оскільки робота, пов‘язана з проведенням дослідження та використанням розробленого програмного забезпечення, є роботою категорії Іа (легкі фізичні роботи, що не потребують фізичного навантаження), норми мікроклімату для даної категорії згідно з ДСН 3.3.6.042-99 для різних пір року наведені у таблиці нижче.

Таблиця 5.1 — Оптимальні величини температури, відносної вологості та швидкості руху повітря в робочій зоні виробничих приміщень (для категорії робіт Іа)

Період року	Температура повітря, °С	Відносна вологість, %	Швидкість руху повітря, м/с
Холодний період	22-24	60-40	0,1
Теплий період	23-25	60-40	0,1

Показники мікроклімату не повинні виходити за межі допустимих значень, що наводяться у наступній таблиці.

Таблиця 5.2 — Допустимі величини температури, відносної вологості та швидкості руху повітря в робочій зоні виробничих приміщень (для категорії робіт Іа)

Період року	Температура повітря, °С				Відносна вологість на постійних та непостійних робочих місцях, %	Швидкість руху повітря на постійних та непостійних робочих місцях, м/с
	Верхня межа		Нижня межа			
	на постійних робочих місцях	на непостійних робочих місцях	на постійних робочих місцях	на непостійних робочих місцях		
Холодний період	25	26	21	18	75	не більше 0,1
Теплий період	28	30	22	20	55 при 28°С	0,2 - 0,1

5.1.2 Небезпечні виробничі фактори

До небезпечних факторів впливу на користувача, пов‘язаних з використанням ЕОМ для проведення дослідження, можна віднести такі:

– нестабільна подача електроенергії;

- перегрівання електронного обладнання ЕОМ;
- надлишковий шум;
- завищена або занижена температура повітря робочої зони;
- завищена або занижена вологість повітря робочої зони;
- недостатня або надлишкова вентиляція приміщення;
- недостатнє або надлишкове освітлення;
- некомфортне розташування та організація робочого місця;
- ризики виникнення порушень роботи організму людини та хвороб, пов‘язаних з діяльністю (порушення зору та опорно-рухового апарату, синдром зап‘ястного тунелю та ін.);
- психофізіологічні фактори (монотонність роботи та малорухливість робітників);
- ненормовані умови праці та графік робіт.

Ризики виникнення професійних хвороб є присутніми навіть, якщо повною мірою дотримуються умови облаштування робочого місця та порядку роботи за ним. Це пов‘язано основним чином з особливостями діяльності осіб, залучених до проведення досліджень у галузі інформаційних технологій — малорухлива робота у постійному положенні сидячи, взаємодія з пристроями з електромагнітним випромінюванням. Окрім того, під час роботи на оператора екранного пристрою також покладається значне інтелектуальне навантаження, що може призвести до розладів психічного здоров‘я. Керівник робіт повинен насамперед провести лабораторні дослідження з умов праці з метою виявлення шкідливих і небезпечних факторів виробничого середовища, важкості та напруженості трудового процесу, а також вживання заходів щодо їх ліквідації [25]. В залежності від впливу факторів робочого середовища на здоров‘я працівників керівником робіт визначається необхідність та періодичність проведення медичних оглядів за власний рахунок, а також регламент порядку виконання робіт, що включає розробку гнучкого графіку роботи з наявністю щоденних перерв та вихідних у відповідності до ст. 50-84 Кодексу законів про працю України [28].

5.1.3 Основні вимоги безпеки

Перелік мінімальних вимог безпеки до екранних пристроїв наведено у НПАОП 0.00-7.15-18 [22] та містить наступні пункти, окрім деяких, що вказані вище:

- усе електромагнітне випромінювання, за винятком видимої частини електромагнітного спектра, має бути зведене до незначного рівня відповідно до Державних санітарних норм і правил захисту населення від впливу електромагнітних випромінювань ДСН 239-96 [27];
- символи на екранних пристроях мають бути чіткими та відповідного розміру. Між символами і рядками символів має бути належна відстань;
- зображення на екрані повинно бути стабільним, без миготінь або інших видів нестабільності;
- яскравість та/або контрастність символів повинна з легкістю регулюватися працівником під час роботи з екранними пристроями, а також швидко адаптуватися до навколишніх умов;
- при виборі екранів слід надавати перевагу таким екранам, які легко та вільно повертаються і нахиляються відповідно до потреби працівника. За необхідності можливо використовувати окрему підставку або регульований стіл для розміщення екрана;
- при виборі клавіатури слід надавати перевагу такій клавіатурі, що відкидається і є відокремленою від екрана, для того, щоб працівник міг вибрати зручну робочу позу й уникнути втоми рук та окремо кисті і верхньої частини руки;
- поверхня клавіатури повинна бути матовою, щоб уникнути віддзеркалювання. Розташування клавіш і самі клавіші мають полегшувати роботу користувача із клавіатурою. Позначення клавіш повинно бути достатньо контрастним і розбірливим.

Зазначеними нормативами не обмежуються права роботодавця на впровадження більш жорстких умов забезпечення охорони праці та безпеки на робочому місці за потреби, якщо ці умови не суперечать чинному законодавству.

5.2 Дії працівників (персоналу) в аварійних (надзвичайних) ситуаціях

Відповідно до Кодексу цивільного захисту України [21] під надзвичайними ситуаціями розуміється обстановка на певній території або господарчому об‘єкті, яка характеризується порушенням нормальних умов життєдіяльності населення внаслідок аварії або іншої небезпечної події, що призвела до виникнення загрози життю або здоров‘ю, великої кількості загиблих і постраждалих, завдання значних матеріальних збитків та до неможливості проживання та ведення господарської діяльності населення на цій території чи об’єкті. Під аварійними ситуаціями розуміється стан потенційно небезпечного об'єкта, що характеризується порушенням меж та/або умов безпечної експлуатації, але не перейшов в аварію, при якому всі несприятливі впливи джерел небезпеки на персонал, населення та навколишнє середовище утримуються у прийнятних межах за допомогою відповідних технічних засобів, передбачених проектом [26]. Більшість аварійних ситуацій, що можуть виникнути під час проведення даного дослідження, в цілому стосуються взаємодії електрообладнання та його оператора з електрострумом. Можливість виникнення таких ситуацій завжди присутня, однак ризик їх виникнення під час експлуатації та поза нею потрібно зводити до мінімуму. Необхідні вимоги щодо дотримання безпеки на робочому місці під час використання екранних пристроїв наведені у відповідних нормативах, що будуть розглянуті нижче.

5.2.1 Правила та вимоги безпечної експлуатації робочого обладнання

У обов‘язковому порядку для мінімізації ризику виникнення аварійної або надзвичайної ситуації, що може бути спричиненою експлуатацією робочого обладнання, на робочому місці повинні дотримуватись вимоги щодо безпечної експлуатації у відповідності до діючого законодавства України.

За Правилами безпечної експлуатації електроустановок споживачів ДНАОП 0.00-1.21-98 [24] робоча станція та її складові, а також засоби захисту під час експлуатації робочого місця повинні бути справними та випробуваними відповідно до вимог чинних нормативних документів. Забезпечення утримання, експлуатації та обслуговування робочих місць відповідно до цих вимог є обов‘язком керівника ро-

біт. Для цього керівник повинен забезпечити виконання переліку обов‘язкових дій, до якого входять:

- призначення особи, відповідальної за справний стан та безпечної експлуатації робочого місця та його компонентів, що має належну підготовку та пройшла перевірку знань (особа, відповідальна за господарство);
- встановлення порядку, за яким працівники, які мають обов‘язки з обслуговування електроустановок, повинні вести ретельне спостереження за дорученим обладнанням — огляд, перевірка дії, випробування та вимірювання;
- проведення протиаварійних, приймально-здавальних і профілактичних випробувань робочого місця відповідно до чинних норм;
- проведення навчальних робіт та перевірки знань з правил технічної експлуатації серед працівників та недопущення до роботи працівників, що не пройшли перевірку знань.

Перелік мінімальних вимог до безпеки під час використання екранних пристроїв наведений у НПАОП 0.00-7.15-18 [22] та складається з наступних пунктів:

1. Щодня перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень (для очищення апаратних складових робочого місця використовуються спеціальні засоби на основі спирту).
2. Після закінчення роботи екранні пристрої слід відключати від електричної мережі.
3. У разі виникнення аварійної ситуації необхідно негайно відключити екранний пристрій від електричної мережі.
4. Не допускається:
    a. виконувати технічне обслуговування, ремонт і налагодження екранних пристроїв безпосередньо на робочому місці працівника під час роботи з екранними пристроями;

б. відключати захисні пристрої, самочинно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;

в. працювати з екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані та інші несправності.

5. Під час виконання робіт операторського типу, пов'язаних з нервово-емоційним напруженням, у приміщеннях під час роботи з екранними пристроями, на пультах і постах керування технологічними процесами та в інших приміщеннях мають дотримуватися оптимальні умови мікроклімату відповідно до вимог ДСН 3.3.6.042-99 (див. табл. 5.1).

5.2.2 Правила та вимоги пожежної та електробезпеки

Відповідно до вимог Правил пожежної безпеки в Україні НАПБ А.01.001-2014 [29] керівник підприємства є особою, відповідальною за проведення організаційних заходів з забезпечення пожежної безпеки, до яких відносяться

- призначення відповідальних осіб за пожежну безпеку приміщення та технічного устаткування;
- встановлення на об'єкті протипожежного режиму відповідними документами, зокрема, порядку відключення від мережі електроживлення обладнання та вентиляційних систем у разі пожежі, порядку організації та експлуатації протипожежних засобів та порядку огляду й зачинення приміщень після закінчення роботи;
- проведення протипожежних інструктажів з працівниками;
- визначення порядку дій у разі настання пожежі згідно з розділом VIII Правил та з урахуванням дотримання техніки безпеки.

До загальних вимог пожежної безпеки до утримання приміщень та обладнання у Правилах, що можна застосувати до офісних приміщень для проведення дослідницьких робіт з замірів затримки обробки переривань у системах реального часу, можливо віднести наступні пункти:

- керівник об‘єкта або підприємства своїм розпорядчим документом визначає спеціальні місця для куріння, які необхідно позначити відповідним знаком або написом, і місця, де встановлюють урну або попільницю з негорючих матеріалів;
- електричні машини, апарати, обладнання, електропроводи та кабелі за виконанням та ступенем захисту повинні мати апаратуру захисту від струмів короткого замикання та інших аварійних режимів;
- електрощити, групові електрощитки повинні оснащуватися схемою підключення споживачів з пояснювальними написами і вказаним значенням номінального струму апарата захисту (плавкої вставки);
- перед початком опалювального сезону теплові мережі, які розташовані у приміщеннях, котельні, теплогенераторні й калориферні установки, печі та інші опалювальні прилади мають бути перевірені й відремонтовані. Несправні опалювальні пристрої не повинні допускатися до експлуатації [29].

Заходи з організації та експлуатації засобів протипожежного захисту повинні відповідати розділу V «Вимоги до утримання технічних засобів протипожежного захисту» Правил пожежної безпеки, зокрема, ДБН В.2.5-56:2014 «Системи протипожежного захисту», та Правилам експлуатації та типові норми належності вогнегасників НАПБ Б.01.008-2018 [29].

Вимоги щодо додержання електробезпеки описуються у Правилах безпечної експлуатації електроустановок споживачів ДНАОП 0.00-1.21-98 [24]. Серед наведених у Правилах пунктів з електробезпеки зазначені наступні вимоги, які можливо застосувати до ЕОМ в офісних приміщеннях:

- машини і пристосування, що застосовуються в електроустановках, повинні бути справні і випробувані відповідно до чинних нормативних документів і строків;
- електрообладнання, конструкції, комплектувальні деталі, вузли вітчизняного та іноземного виробництва повинні відповідати вимогам чинних в Україні нормативних документів;

– до оперативного обслуговування електроустановок (у даному випадку — обслуговування та ремонт ЕОМ) допускаються працівники, які знають оперативні схеми, посадові і експлуатаційні інструкції, інструкції з охорони праці, особливості обладнання і пройшли навчання, дублювання та перевірку знань цих Правил та правил технічної експлуатації (ПТЕ) ЕОМ.

5.2.3 Регламент дій у разі настання аварійних (надзвичайних) ситуацій

Якщо на робочому місці, екранному пристрої або його окремих складових компонентах під час тестових випробувань або постійної експлуатації були віднайдені хоча б найменші ознаки несправності (пошкодження, іскріння, поява диму та/або його відчутного запаху тощо), або якщо термін експлуатації дійшов до кінця, слід у першу чергу негайно припинити його використання [22] та звернутись до керівника робіт або особи, відповідальної за господарство.

У разі, якщо сталося аварійне відключення електроенергії, необхідно негайно відключити обладнання робочої станції від електромережі.

Якщо у приміщенні розташування робочих місць виникла будь-яка аварійна або надзвичайна ситуація, що перешкоджає безпечному вимкненню електрообладнання (пожежа, потоп, землетрус та ін.), потрібно вжити заходів щодо повідомлення відповідної рятувальної служби та евакуації працівників (за попередньо розробленим планом) з приміщення. У разі, якщо пожежа була виявлена передчасно, потрібно вжити заходів щодо її гасіння наявними первинними засобами пожежогасіння.

У разі виявлення постраждалих від наслідків виникнення аварійної або надзвичайної ситуації потрібно спершу при необхідності евакуювати їх з приміщення, а потім надати їм первинну медичну допомогу в залежності від типу ураження та викликати швидку допомогу.

АНАЛІЗ РЕЗУЛЬТАТІВ ТА ЗАГАЛЬНІ ВИСНОВКИ

Під час проведення даного дослідження було розглянуто три основні випадки затримки обробки переривань, що були змодельовані для розробки програмного забезпечення автоматизації проведення розрахунків, а також визначено їх вплив на статистичні показники, отримані у результаті виконання експериментальних замірів часу затримки. Крім того, було розглянуто вплив на результуючі дані таких факторів, як використання віртуальної машини та навантаження процесами операційної системи, у середовищі якої виконувались розроблені програми обробки переривань. За результатами проведення досліджень часових характеристик затримки обробки переривань було сформульовано ряд висновків, які представлені нижче.

Тестування затримки переривання на віртуальній ЕОМ у порівнянні з фізичною, або чистою ЕОМ, у результаті дало більші за значеннями статистичні показники — різниця між показниками обох машин знаходиться у діапазоні 52—98%. Окрім більшого значення математичного очікування, для даних, отриманих з віртуальної машини, також спостерігаються значні відхилення (до 86%) даних заміру затримки. Це свідчить про те, що віртуалізація має значний вплив на часові дані та вносить погрішність, якою неможливо нехтувати, тому для тестування програмного забезпечення, що розробляється для ОСРЧ QNX, важливо використовувати фізичну систему з метою отримання більш ефективних та точних результатів.

Вплив високого рівня навантаження системи, у свою чергу, показав відносно незначний вплив на результуючі часові дані (збільшення статистичних показників у діапазоні 0,02—6%), що пояснюється особливостями QNX як операційної системи реального часу з мікроядерною архітектурою, тобто, архітектурою, що складається з мікроядра та процесів, які існують незалежно від нього. Тому існує можливість нехтування додатковим часом, спричиненим затримкою для деяких видів переривань (наприклад, переривання контролерів жорсткого диску).

При порівнянні затримок звичайного та відкладеного потоків обробки було відмічено дуже значну різницю у показниках (затримка відкладеного потоку менша за звичайний потік майже у 10 разів для усіх розглянутих джерел переривань). Таку ситуацію можливо пояснити тим, що для запуску потоку звичайної обробки відбува-

ється переключення контексту з користувацького потоку до ядра, на час виконання якого мають вплив інші потоки та процеси, що функціонують у тому самому середовищі та потребують обробки своїх сигналів та повідомлень, а також швидкість отримання потоком сигналу переривання, яка залежить від типу джерела переривання та його фізичній конфігурації (на прикладі досліджених затримок звичайного планування потоків обробки переривань PS/2-клавіатури та USB-миші для фізичної системи помітно, що протокол USB забезпечує більш швидку доставку сигналу переривання до системи з різницею майже у 40%). Для запуску відкладеного потоку відбувається переключення з потоку ядра, яке відповідно до реалізації архітектури QNX завжди має найвищий пріоритет, на новий користувацький потік. Саме високий пріоритет потоку обробки переривання має вплив на суттєве зменшення часових показників.

Обробка переривань за пріоритетністю показала досить неочікувані результати. На початку дослідження припускалось, що потік обробки переривання з вищим пріоритетом суттєво збільшить часові показники паралельного потоку обробки переривання з нижчим пріоритетом, однак це припущення виявилось лише частково правильним — з паралельним запуском потоку переривання вищого пріоритету значення дисперсії та стандартного відхилення потоку нижчого переривання збільшилось на більш, ніж 90%, у той час, як математичне очікування зазнало незначного зменшення майже на 2%. Використання відкладеної обробки для потоку переривання нижчого пріоритету показало обернену тенденцію у різниці показників — збільшення математичного очікування майже на 0,3% та зменшення стандартного відхилення майже на 50%. Втім, оскільки результуючі дані часових замірів не перевищують 1 мкс (за винятком аномального значення максимуму у 6 мкс для самостійного функціонування потоку низького пріоритету), для деяких задач м‘якого реального часу цими показниками також можливо знехтувати.

Досліджений вплив затримки обробки переривання на значення WCET показав залежність часових результатів від реалізації програми, для якої досліджується WCET, та від типу джерела переривання, обробка якої виконується у програмі. Для елементарних програм обробки переривань затримка обробки займає майже увесь

час їх функціонування (97,4—99,9% від загального часу у мкс), а для складних програм помітна залежність часу від типу джерела переривання — час на обробку переривань контролеру HDD не перевищує 0,6% від загального часу, обробка переривань системного таймеру займає у середньому близько 7% загального часу, а частка загального часу на обробку переривань клавіатури знаходиться у діапазоні 83—90%.

Отримані результати досліджень та їх пояснення не можна назвати остаточними. Причиною цьому є обмеженість використаних ЕОМ у апаратному та програмному контекстах, що застосовувались під час експериментальних замірів часу — цілком можливо, що для інших версій ОСРЧ QNX та ЕОМ з іншою апаратною складовою результати будуть дещо відмінними від отриманих. Ще однією причиною є огляд проблеми виключно з програмної точки зору, оскільки не виконувалось жодних досліджень та модифікацій апаратних складових, включаючи периферійні пристрої, однак, як було визначено у вступній частині, огляд переривань та їх обробки у рамках проведення даного дослідження проводиться лише з програмної точки зору, апаратна складова залишається для інших, окремих дослідницьких робіт.

Проведені розрахунки дозволять уточнити та доповнити загальну оцінку метрики WCET на програмному рівні для класів задач реального часу, правильний та вчасний розв‘язок яких залежить від часу обробки переривань системою.

На основі розглянутих джерел інформації та спостережень під час дослідження проблеми був сформульований список наступних рекомендацій та вимог до програм, що використовують переривання та їх обробку як засіб комунікації з програмним та апаратним оточенням:

– через можливість перевищення часових затрат системи на переключення контексту під час виклику функції обробки переривання об‘єм операцій, виконуваних всередині функції обробки переривання, повинен зводитись до мінімуму — наприклад, для маніпулювання регістрами пристроїв доцільно використовувати побітові операції, представлені у бібліотеці функцій QNX Neutrino як системні виклики in* та out*, а для швидких арифметичних операцій слід використовувати функції бібліотеки atomic;

- усі змінні, що використовуються всередині функцій обробки переривань, повинні захищатись за допомогою їх оголошення з використанням ключового слова volatile [5] — це дасть змогу компіляторові уникнути кешування їх значення та, як результат, отримання некоректних даних після завершення обробки;
- під час формування висновків на основі тестування програм слід надавати перевагу результатам, отриманим на фізичній машині з ОРСЧ QNX, якщо є можливість її використання;
- затримка переривань має значно більший вплив на метрику WCET для задач, розв‘язок яких залежить від комунікацій з периферійними пристроями, тому при диспетчеризації потоків обробки переривань периферійних пристроїв слід приділяти особливу увагу цим пристроям та взаємодії програми з ними як на програмному, так і на апаратному рівнях.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. QNX Neutrino Realtime Operating System: Programmer’s Guide For QNX Neutrino 6.3.2 [Електронний ресурс] / QNX Software Systems GmbH & Co. KG — 2007 — Режим доступу: https://www.qnx.com/developers/docs/6.3.2/neutrino/prog/about.html
2. QNX Neutrino Realtime Operating System: Programmer’s Guide For QNX Neutrino 7.1.0 [Текст] / QNX Software Systems Limited, a subsidiary of BlackBerry. — 2020.
3. Hildebrand, D. An Architectural Overview of QNX [Текст] / D. Hildebrand / Quantum Software Systems Ltd. — 1992 — Режим доступу: https://cseweb.ucsd.edu/~voelker/cse221/papers/qnx-paper92.pdf
4. Махилёв, В. Результаты тестов производительности QNX Neutrino [Текст] / В. Махилёв // Журнал «Современные технологии автоматизации» — 2012. — №2. — С. 82-89.
5. Кёртен, Роб. Введение в QNX Neutrino 2 / Р. Кёртен. — В переводе А. Н. Алексеева / М: Петрополис, 2001 — 480 стр.
6. Никольский, О.Л. Программирование приложений реального времени для исполнения в среде операционной системы QNX/Neutrino, часть II. Обработка прерываний в операционной системе реального времени QNX/Neutrino / О. Л. Никольский — Москва: БГТУ, 2007
7. Bogani, A., Kacur. J. cyclictest (8) manual page [Електронний ресурс] / Alessio Igor Bogani, John Kacur // ArchLinux System Manager's Manual. — Режим доступу: https://man.archlinux.org/man/
8. cyclictest.8. — Назва з екрана. — Дата останнього оновлення: 22.04.2016. — Дата перегляду: 12.10.2021.
9. Grandegger, W. gpioirqbench Readme [Електронний ресурс] / Wolfgang Grandegger // DENX Software Engineering. — Режим доступу: http://www.denx.de/wiki/DULG/AN2008_03_Xenomai_
10. gpioirqbench. — Назва з екрана. — Дата публікації: 07.03.2008. — Дата перегляду: 12.10.2021.

11. QNX® Software Development Platform 6.6 System Analysis Toolkit (SAT) User's Guide [Текст] / QNX Software Systems Limited, a subsidiary of BlackBerry. — 2014 — 20 February
12. Walls, C. Measuring RTOS interrupt latency [Електронний ресурс] / Colin Walls, Siemens Embedded // Youtube. — Режим доступу: https://www.youtube.com/watch?v=bjXBQE6TDo4. — Дата публікації: 29.11.2018. — Дата перегляду: 01.08.2021.
13. QNX Neutrino Realtime Operating System: Library Reference For QNX Neutrino 6.3.2 [Електронний ресурс] / QNX Software Systems GmbH & Co. KG — 2007 — Режим доступу: https://www.qnx.com/developers/docs/6.3.2/neutrino/lib_ref/about.html
14. Теорія ймовірностей та математична статистика: Частина 2. Випадкові величини: Лекції і практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 143 «Атомна енергетика», спеціалізації «Атомні електричні станції» / КПІ ім. Ігоря Сікорського; уклад.: І. В. Веригіна, О. В. Островська. Київ : КПІ ім. Ігоря Сікорського, 2021. — 77 с.
15. Гмурман В. Е. Руководство к решению задач по теории вероятностей и математической статистике / В. Е. Гмурман — М.: Высш. школа, 1979. — 400 стр.
16. Математическая статистика [Текст] : учебное пособие / Д. К. Агишева, С. А. Зотова, Т. А. Матвеева, В. Б. Светличная. — ВолгГТУ, 2010. — 160 с.
17. Stecher, B. The documentation is wrong. ClockCyles() is safe to use in all environments on all architectures. The kernel instrumentation wouldn't work if that wasn't true. [Електронний ресурс] / Brian Stecher // Форум foundry27 — Режим доступу: https://community.qnx.com/sf/go/topc2214. — Дата публікації: 29.11.2018. — Дата перегляду: 22.09.2021.
18. Watt, David A. Programming language concepts and paradigms [Текст] / David A. Watt. — New York: Prentice Hall, 1990 — 322 с.

19. Арлоу Д., Нейштадт И. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2е издание. – Пер. с англ. Н. Шатохиной – СПб: Символ Плюс, 2007. – 624 с.
20. Patton, Ron. Software Testing [Текст] / Ron Patton. — Indianapolis, IN : Sams Pub., 2006 — 420 с.
21. Охорона праці. Методичні рекомендації до розробки розділу дипломних проектів бакалаврам.[Текст]: уклад.: В. Г. Лоза; Дніпровський нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. — Дніпро, 2020. — 28 с.
22. НПАОП 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями.
23. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень.
24. ДНАОП 0.00-1.21-98. Правила безпечної експлуатації електроустановок споживачів.
25. Кодекс цивільного захисту України.
26. ДНАОП 0.00-4.33-99. Положення щодо розробки планів локалізації та ліквідації аварійних ситуацій і аварій.
27. ДСН 239-96. Державні санітарні норми і правила захисту населення від впливу електромагнітних випромінювань.
28. Кодекс законів про працю України.
29. НАПБ А.01.001-2014. Правила пожежної безпеки в Україні.

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Борис БОДНАР
01.09.2021

ПРОГРАМНИЙ КОМПЛЕКС «QNX ISR TEST TOOLKIT»

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

01116130.01197-01-ЛЗ

Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
01.09.21

Керівник розробки
Віктор НЕЧАЙ
01.09.21

Виконавець
Алла ВОЛКОДАВЕЦЬ
01.09.21

Нормоконтролер
Олена КУРОП‘ЯТНИК
01.09.21

2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

0111630. 01197-01

ПРОГРАМНИЙ КОМПЛЕКС «QNX ISR TEST TOOLKIT»

Технічне завдання

Аркушів 23

2021

ЗМІСТ

ВСТУП

Програмний комплекс «QNX ISR Test Toolkit» являє собою набір утиліт, призначених для дослідження часових характеристик затримки обробки переривань в операційній системі реального часу QNX Neutrino на базі експериментальних та статистичних даних. Утиліти даного програмного комплексу моделюють три основні ситуації, під час яких можуть виникнути затримки запуску ядром системи потоків, відповідальних за обробку системних та апаратних переривань, а також фіксують та розраховують часові дані щодо цієї затримки.

Необхідність розробки даного програмного забезпечення обумовлена недостатністю викладення методології проведення аналогічних досліджень в існуючих статтях та наукових роботах, присвячених перериванням у ОСРЧ QNX та їх програмній обробці.

Програмний комплекс може застосовуватись у будь-якій галузі, що потребує розробки програмного забезпечення розв‘язку задач жорсткого часу, що використовують переривання як один з методів взаємодії з навколишнім середовищем (наприклад, через периферійні пристрої або електронні датчики), для отримання інформації про значення метрики найгіршого часу виконання (WCET) програми, що розробляється.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ ректора Дніпровського національного університету залізничного транспорту ім. академіка В. Лазаряна Пшінька О. М. №690 ст. від 18.11.2020 р. «Про призначення керівників та затвердження тем магістерських робіт» за спеціальністю 121 «Інженерія програмного забезпечення» факультету «Комп‘ютерних технологій і систем».

Тема дипломної роботи — «Дослідження часових характеристик затримки переривань у системах реального часу».

Керівник дипломної роботи — Нечай В. Я.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення програми полягає у виконанні багатократних замірів часу затримки обробки переривань та їх статистичної обробки, а саме підрахунку значень математичного очікування, дисперсії та стандартного відхилення.

Експлуатаційне призначення програми полягає у наданні наглядного статистичного аналізу часових характеристик затримки обробки переривань різного типу в операційній системі реального часу QNX 6 для отримання більш повних результатів замірів метрики WCET.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик наступні:

- забезпечення моделювання трьох основних ситуацій виникнення затримки обробки переривань у ОСРЧ QNX: просте планування ядром потоку-обробника, використання відкладених переривань та затримка за пріоритетністю;
- багатократна фіксація часу затримки окремо для кожного розглянутого випадку.

Вхідними даними є символьні параметри та їх значення — номери лінії переривань для досліджень, час затримки початку роботи програми та часом виконання замірів у секундах, необхідність запису результатів у файл.

Вихідними даним є результати підрахунку статистичних метрик та експериментальні заміри у текстовому вигляді, а також, якщо було встановлено параметр збереження даних у файл, файл формату .log з експериментальними даними.

3.2 Вимоги до надійності

Вимоги до надійності наступні:

- відсутність небезпечних системних викликів у потоках ядра обробки переривань;
- інкапсуляція даних у програмних класах;
- перевірка на помилки у результаті роботи системних викликів та надання користувачеві відповідних повідомлень у разі їх виникнення;
- наявність архівної копії програми на фізичному носії.

3.3 Умови експлуатації

Для забезпечення коректного функціонування програмного продукту необхідно дотримуватись таких умов:

- програмний продукт повинен використовуватись у приміщенні, яке повинно відповідати умовам експлуатації ЕВМ, а саме — мати гігієнічні та санітарні умови, які відповідають Вимогам щодо безпеки та захисту

здоров’я працівників під час роботи з екранними пристроями НПАОП 0.00-1.31-99 [2];

- програмний комплекс повинен функціонувати у приміщеннях, призначених для роботи на ЕОМ з наступними кліматичними умовами: для теплої пори року температура повітря має бути у межах 23-25°С з відносною вологістю повітря 40-60%; для холодної пори року температура — у межах 22-24 °С з вологістю повітря 40-60% згідно з Санітарними нормами мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [3];
- для роботи з програмою користувач повинен мати базові навички роботи з персональним комп’ютером, командною строкою bash та операційною системою QNX, а також бути ознайомлений з керівництвом користувача.

3.4 Вимоги до складу і параметрів технічних засобів

Програма, що розробляється, розрахована для використання на IBM-сумісних комп’ютерах, що мають наступні мінімальні характеристики:

- процесор — мінімум одноядерний 32-, 64- або 86-бітний процесор архітектури x86;
- оперативна пам’ять — мінімум 256 МБ;
- вільна пам’ять на жорсткому диску — мінімум 350 КБ;
- периферійні пристрої — VGA-монітор з мінімальним розширенням екрану 800x600, клавіатура, миша, дисковод компакт-дисків.

3.5 Вимоги до інформаційної і програмної сумісності

Для роботи з програмою на ПК має бути встановлене наступне програмне забезпечення:

- ОС — операційна система реального часу QNX Neutrino версії 6 або пізніше;
- програми-драйвери периферійних пристроїв, переривання яких будуть тестуватися даною програмою.

Програмний продукт має бути розроблений з використанням мови програмування C++, включаючи сумісні з нею системні виклики ядра ОСРЧ QNX для забезпечення взаємодії системи з сигналами апаратних та програмних переривань.

3.6 Вимоги до маркування і упаковки

У разі необхідності випуску даного програмного продукту у фізичному форматі слід притримуватись наступних вимог:

- програмний продукт, а також електронна документація користувача повинні зберігатись на диску типу CD-ROM для перегляду на комп’ютері;
- вміст упаковки повинен включати паперову копію документації з користування, а також паперові копії ліцензій на використання самої програми та додаткових програмних модулів, використаних при розробці, та гарантійний талон зі вказівкою повної контактної інформації про розробника;
- вміст упаковки повинен бути поміщений у пластиковий контейнер, який, в свою чергу, поміщається в коробку з цупкого картону, на зовнішній стороні якої повинні бути розташовані логотип програмного продукту, короткий опис та маркування, як показано на рис. 3.1 нижче.

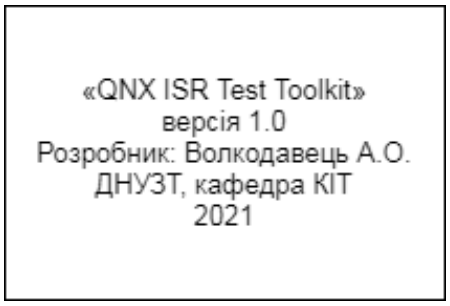

Рисунок 3.1 — Приклад маркування упаковки

3.7 Вимоги до транспортування і зберігання

Транспортування фізичних копій повинно забезпечувати збереження програмного продукту, його цілісність та запобігання несанкціонованого доступу до нього. Транспортування фізичної упаковки програмного продукту повинно

проводитись довіреною юридичною особою та здійснюватися комплектами в упаковці з термоплівки та піни, яка захищає носій від різного виду пошкоджень. Місце зберігання програмного продукту повинне бути сухим з відсутністю пилу при температурі 21-25°С і відносної вологості 40-60% [3], з уникненням впливу вологи та шкідників.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація представляє собою перелік наступних документів:

1. технічне завдання;
2. робочий проект, що містить такі документи:
    а. специфікація;
    б. текст програми;
    в. опис програми;
    г. керівництво користувача. Керівництво з використання програмного комплексу «QNX ISR Test Toolkit».

Оформлення усієї документації повинно відповідати вимогам і нормам ГОСТ 19.106-78 «Єдина система програмної документації» [4].

5 ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Техніко-економічні показники мають на меті надання фінансової оцінки витрат, що передбачуються на розробку програмного продукту, а також економічної доцільності його розробки та впровадження. У цьому розділі наведено розрахунки деяких економічних показників для проекту розробки експериментального програмного забезпечення виконання замірів часу затримки обробки переривань в операційній системі реального часу QNX.

Оцінка розміру програмного продукту виконана за моделлю оцінки вартості розробки програмного забезпечення COCOMO. Відповідно до даної моделі[5] розмір проекту S визначається в тисячах рядках коду (KLOC) та пов‘язаний з формулою

$$E = a \cdot S^b \cdot EAF, \tag{6.1}$$

де E — витрати праці на проект (в людино-місяцях);

S^b — розмір коду (в KLOC);

EAF — фактор уточнення витрат (effort adjustment factor), приймається як 1;

система, що була розроблена, є простою, тому $a = 2{,}4$; $b = 1{,}05$.

Код програмного комплексу «QNX ISR Test Toolkit», розробленого під час виконання даної дослідницької роботи, складається з чотирьох модулів обчислення часових значень затримок обробки переривань, кожен з яких містить наступну кількість рядків (не враховуючи автоматично створений середовищем розробки код):

- модуль замірів затримки планування — 128 LOC;
- модуль замірів затримки відкладених обробників переривань — 115 LOC;
- модуль замірів затримки обробників за пріоритетом — 163 LOC;
- модуль статистичного аналізу даних — 60 LOC.

Розмір програмного коду загалом складає 466 LOC, або 0,466 KLOC, отже,

$$E = 2{,}4 \cdot 0{,}466^{1{,}05} \cdot 1 = 1{,}076 \approx 1 \text{ людино-місяць}$$

Далі наведені дані щодо розрахунку вартості розробки програмного комплексу «QNX ISR Test Toolkit».

Розрахунок значення метрики основної заробітної плати (ОЗП), що призначена для оцінки праці зі створення програмного продукту, проводиться за середніми даними заробітної плати, що пропонується в оголошеннях, розміщених на веб-сайті work.ua [6] у м. Дніпро станом на 01.11.2021. Оскільки дане дослідження проводиться для операційної системи реального часу, кваліфікація учасників проектної команди повинна бути на рівні розробників для мікроконтролерів, або embedded-розробників, тому значення заробітної плати було обране відповідно до даної кваліфікації.

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати, грн
			чол	місяців	
1	embedded-розробник	16000	2	1	32000

Проект розробки програмного комплексу «QNX ISR Test Toolkit» передбачає формування команди з 2-х осіб та встановлення терміну розробки тривалістю 1 місяць — 30 днів або 4 робочих тижні. Витрати робочого часу приймаються за 40 годин на тиждень. Формула розрахунку витраченого робочого часу —

$$t_{\text{розробки}} = N_{\text{чол}} \cdot N_{\text{тиж}} \cdot N_{\text{год}}, \quad (6.2)$$

де $N_{\text{чол}}$ – кількість виконавців, *чол*;

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, год.

$$t_{\text{розробки}} = 2 \cdot 4 \cdot 40 = 320 \text{ год.}$$

Погодинна ставка у даному випадку дорівнює 100 грн.

Формула розрахунку ОЗП —

$$\text{ОЗП} = t_{\text{розробки}} \cdot N \cdot K_{KB}, \quad (6.3)$$

де $t_{\text{розробки}}$ – витрати праці у чол/год;

N – погодинна ставка;

K_{KB} – коефіцієнт кваліфікації програміста, приймається 0,75.

$$\text{ОЗП} = 320 \cdot 100 \cdot 0{,}75 = 24000 \text{ грн.}$$

За отриманими результатами встановимо відрахування на соціальні потреби. Згідно з чинним Законодавством України нарахування на соціальні потреби становлять 22% від основної заробітної платні.

$$C_{\text{соц}} = \frac{\text{ОЗП} \cdot 22\%}{100\%} = \frac{24000 \cdot 22\%}{100\%} = 5280 \text{ грн.} \quad (6.4)$$

Розрахуємо значення основних прямих витрат на розробку програмного продукту.

Накладні витрати розраховуються за формулою

$$\text{С}_{\text{накл}} = \frac{(\text{ОЗП}+\text{С}_{\text{соц}})\cdot 35\%}{100\%} = \frac{(24000+5280)\cdot 35\%}{100\%} = 10248 \text{ грн.} \quad (6.5)$$

Експлуатаційні витрати визначаються за формулою

$$\text{С}_{\text{ел}} = P \cdot B \cdot T_{\text{розр}}, \quad (6.6)$$

де Р – потужність комп’ютера та допоміжних споживачів електричної енергії, приймається 0,45 кВт/год;

B – вартість 1кВт/год, станом на 01.11.2021 складає 1,44 грн [7];

$T_{\text{розр}}$ – час роботи з ЕВМ, приймається рівним робочому часу.

$$C_{ел} = 0{,}45 \cdot 1{,}44 \cdot 320 \approx 207 \text{ грн.} \quad (6.7)$$

Витрати на витратні матеріали (у даному випадку комплектуючі робочої станції)

$$C_{вм} = B_{КОМ} \cdot \frac{N_Д}{N_{експ} \cdot 365} \cdot \frac{10\%}{100\%}, \quad (6.8)$$

де $B_{КОМ}$ – вартість персонального комп'ютеру;

$N_Д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп'ютеру.

Вартість персонального комп'ютеру $B_{КОМ}$ визначимо як суму середніх вартостей апаратних комплектуючих, наявних у продажі в Україні станом на 01.11.2021 в онлайн-магазині Foxtrot.ua[8], що відповідають поставленим у технічному завданні мінімальним системним вимогам.

Таблиця 5.2 – Вартість комплектуючих персонального комп'ютеру

Найменування	Середня вартість, грн
Процесор	3095
Материнська плата	2888
Модуль оперативної пам'яті	533
SSD-накоплювач	1001
Блок живлення	548
Дисковод DVD-RW	914
Корпус системного блоку	850
Монітор	3839
Клавіатура PS/2	292
Миша PS/2	216
Всього	14176

Термін експлуатації комп'ютеру приймемо за 3 роки.

$$\text{С}_{\text{вм}} = 14173{,}50 \cdot \frac{30}{3 \cdot 365} \cdot \frac{10\%}{100\%} \approx 39 \text{ грн.} \quad (6.9)$$

Значення заробітної плати ремонтника розраховується за формулою

$$C_{\text{рем}} = \frac{C_{\text{рем}}'}{N_{КОМ}} \cdot T_{\text{міс}}, \quad (6.10)$$

де $C_{\text{рем}}'$ – середньомісячна заробітна плата (отримана за даними сайту work.ua[9]);

$N_{КОМ}$ – кількість комп'ютерів на одного ремонтника;

$T_{мес}$ – час розробки програмного продукту, міс.

$$C_{\text{рем}} = \frac{15000}{2} \cdot 1 = 7500 \text{ грн.}$$

Витрати на комплектуючі вироби приймаються за 10% від вартості за термін його експлуатації, тобто, рівною значенню витрат на витратні матеріали.

$$C_{КОМ} = C_{ВМ} = 39 \text{ грн.}$$

Амортизаційні витрати на апаратне забезпечення розраховуються за терміном корисного використання матеріальних активів, що був встановлений як 3 роки. Формула розрахунку —

$$\text{АКП} = \text{В}_{КОМ} \cdot \frac{N_{\text{Д}}}{N_{\text{експ}} \cdot 365} = 14173{,}50 \cdot \frac{30}{3 \cdot 365} \approx 388 \text{ грн.} \quad (6.11)$$

У табл. 5.2 представлені усі нематеріальні активи (програмне забезпечення) для придбання у цілях проведення робіт, а також амортизаційні відрахування за їх вартістю. Так як дослідження затримок обробок переривань було проведено для фізичної та віртуальної машин, проект розробки включає тестування розробленої програми на обох машинах. Фізична машина реалізується компонентами пакету програм QNX Software Development Package 7.1. Віртуальна машина реалізується з використанням програми-гіпервізора VMware Workstation 16 Pro на базі фізичного комп'ютера з операційною системою Microsoft Windows 10 Pro.

Вартість у гривнях була отримана за значенням вартості 1 американського долару за даними фінансового онлайн-сервісу Xe.com [10] станом на 01.11.2021, що становить 26,3069 грн. Термін корисного використання даних нематеріальних активів був встановлений окремо для кожного програмного продукту:

– для QNX Software Development Package 7.1 — 4 календарних роки;
– для Microsoft Windows 10 Pro — 3 календарних роки;
– для VMware Workstation 16 Pro — 5 календарних років.

Результуючі значення амортизаційних відрахувань за один місяць також наведені у таблиці 5.2.

Таблиця 5.2 – Використовуване програмне забезпечення та амортизаційні відрахування

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
QNX Software Development Package 7.1 Commercial License	13153	https://blackberry.qnx.com/en/company/contact	274
Microsoft Windows 10 Pro Digital License	5261	https://www.microsoft.com/en-us/d/windows-10-pro/df77x4d43rkt/48DN?activetab=pivot:overviewtab	146
VMware Workstation 16 Pro	5235	https://store-us.vmware.com/vmware-workstation-16-pro-5424176500.html	87
Всього:	23649		507

Додаткові витрати ($C_{\text{дод}}$), такі, як прибирання приміщень, комунальні та охоронні послуги, приймаються як 50% від заробітної плати embedded-розробника, тобто, 8000 грн/міс.

Оренду приміщень приймемо за даними вартості оренди приміщення в м. Дніпро, що представлені у веб-сервісі оголошень OLX [10] станом на 01.11.2021. Середня вартість оренди за м^2 становить 347,5 грн/міс, отже, вартість оренди приймемо як сумарну вартість оренди приміщення площею 20 м^2, що дорівнює 6950 грн/міс.

Сумарні експлуатаційні витрати на один персональний комп'ютер складають:

$$C_{\text{експ}} = C_{\text{ел}} + C_{\text{ВМ}} + C_{\text{рем}} + \text{АПК} + \text{АПЗ} + C_{\text{ор}} + C_{\text{дод}}; \qquad (6.12)$$

$$C_{\text{експ}} = 207 + 39 + 7500 + 388 + 507 + 6950 + 8000 = 23591 \text{ грн.}$$

Усі отримані результати розрахунків, використані при підрахунку експлуатаційних витрат, наведено в наступній таблиці.

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПЗ

Найменування витрат	Витрати, грн
Витрати на електроенергію	207
Вартість витратних матеріалів	39
Витрати на ремонт	7500
Амортизація персонального комп'ютера	388
Амортизація програмного забезпечення	507
Оренда приміщення	6950
Додаткові витрати	8000
Всього	23591

Таким чином, витрати на створення програмного продукту складають

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}; \quad (6.13)$$

$$C_{\text{розробки}} = 24000 + 5280 + 10248 + 23591 = 63119 \text{ грн.}$$

Результуючий розрахунок витрат на розробку зведено у кошторисі, представленому таблицею 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	24000
Відрахування на соціальні потреби	5280
Накладні витрати	10248
Експлуатаційні витрати	23592
Всього	63120

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Розробка програмного продукту планується проводитись ітеративним методом організації життєвого циклу проекту. План розробки, що складає одну ітерацію загального процесу, складається з чотирьох основних стадій та виглядає наступним чином:

- проектування програмного модулю;
- розробка програмного модулю згідно з проектом;
- тестування і відлагодження модулю;
- впровадження модулю в цілісний програмний продукт.

Відповідно до загального плану розробки, модулі, що повинні бути створеними у результаті виконання кожної ітерації, є наступними:

- модуль замірів часу на планування обробки переривань;
- модуль замірів часу затримки обробки відкладених переривань;
- модуль замірів часу затримки функцій обробки переривань з низьким пріоритетом.

Документи, які необхідно розробити та затвердити на кожному етапі, зазначені у нижче наведеному переліку.

1. технічне завдання;
2. робочий проект у складі таких документів:
    а. специфікація;
    б. текст програми;
    в. опис програми;
    г. керівництво користувача. Керівництво з використання програмного комплексу «QNX ISR Test Toolkit».

Нижче приведена таблиця, яка описує усі заплановані стадії розробки програми.

Таблиця 6.1 — Стадії процесу розробки

Стадія	Зміст робіт	Терміни виконання
1 Технічне завдання	– виокремлення та проектування основних програмних модулів – аналіз необхідних витрат на розробку – планування робіт – узгодження і затвердження технічного завдання	06.09.2021 — 17.09.2021
2 Робочий проект	– написання програмного коду – оформлення програмного інтерфейсу – написання програмної документації	20.09.2021 — 13.10.2021
3 Розробка програмного комплексу	– складання плану тестування – виконання тестування програмних модулів та їх роботи разом – складання звітності з результатів тестування – відлагодження програми за результатами звіту	18.10.2021 — 25.10.2021
4 Розгортка та прийом	– розробка, узгодження і затвердження програмної документації	26.10.2021 — 01.11.2021

7 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль за виконанням роботи виконує науковий керівник дипломного проекту — Нечай В.Я.

Прийом готового програмного продукту здійснюється уповноваженою комісією.

8 БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. тра-нсп. ім. акад. В. Лазаряна, 2009. - 38 с.
2. НПАОП 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями.
3. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень.
4. Інженерія програмного забезпечення [Текст] : навчальний посібник / В. І. Шинкаренко, О. В. Горбова, О. П. Іванов, В. О. Андрющенко, В. Я. Нечай; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Дніпро, 2019. – 140 с.
5. Software Engineering. COCOMO model. // Портал GeeksforGeeks. Дата оновлення: 08 червня 2020. URL: https://www.geeksforgeeks.org/software-engineering-cocomo-model/ (дата звернення: 17.11.2021).
6. Робота: embedded-developer у Дніпрі. Вакансії і робота — Work.ua // Сервіс пошуку роботи Work.ua. URL: https://www.work.ua/jobs-dnipro-embedded+developer/ (дата звернення: 01.11.2021).
7. Недогибиченко, А. Нові тарифи на електроенергію. Скільки платитимемо з 1 жовтня // Український фінансовий портал Minfin.com.ua. Дата оновлення: 1 жовтня 2021. URL: https://minfin.com.ua/ua/2021/10/01/72689923/ (дата звернення: 01.11.2021).
8. Компьютерные комплектующие — купить в интернет-магазине ФОКСТРОТ // Інтернет-магазин техніки ФОКСТРОТ. URL: https://www.foxtrot.com.ua/ru/portal-komputernie-komplektuushie.html(дата звернення: 01.11.2021).

9. Робота: ремонтник у Дніпрі. Вакансії і робота — Work.ua // Сервіс пошуку роботи Work.ua. URL: https://www.work.ua/jobs-dnipro-%D1%80%D0%B5%D0%BC%D0%BE%D0%BD%D1%82%D0%BD%D0%B8%D0%BA/(дата звернення: 01.11.2021).
10. 1 USD to UAH — US Dollars to Ukrainian Hryvni Exchange Rate — Xe Currency Converter // Сервіс конвертації валют Xe.com. URL: https://www.xe.com/currencyconverter/convert/?Amount=1&From=USD&To=UAH (дата звернення: 01.11.2021).
11. Аренда офиса Днепр, снять офис — объявления OLX.ua Днепр // Сервіс оголошень OLX. URL: https://www.olx.ua/nedvizhimost/kommercheskaya-nedvizhimost/arenda-kommercheskoy-nedvizhimosti/ofisnye-pomeschenіya/dnepr/(дата звернення: 01.11.2021).

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Дніпровського
національного університету
залізничного транспорту
імені академіка В. Лазаряна
Борис БОДНАР
01.09.2021

ПРОГРАМНИЙ КОМПЛЕКС «QNX ISR TEST TOOLKIT»

Робочий проект

ЛИСТ ЗАТВЕРДЖЕННЯ

01116130.01197-01-ЛЗ

Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
01.09.21

Керівник розробки
Віктор НЕЧАЙ
01.09.21

Виконавець
Алла ВОЛКОДАВЕЦЬ
01.09.21

Нормоконтролер
Олена КУРОП‘ЯТНИК
01.09.21

2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

0111630. 01197-01

ПРОГРАМНИЙ КОМПЛЕКС «QNX ISR TEST TOOLKIT»

Специфікація

0111630. 01197-01

Аркушів 2

2021

Позначення	Найменування	Примітка
01116130.01197-01	Специфікація	
01116130. 01197-01 12 01	Текст програми	
01116130. 01197-01 13 01	Опис програми	
01116130.01197-01 ІЗ 01	Керівництво користувача. Керівництво з використання програмного комплексу «QNX ISR Test Toolkit»	

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

01116130.01197-01 13 01

ПРОГРАМНИЙ КОМПЛЕКС «QNX ISR TEST TOOLKIT»

Опис програми

01116130.01197-01 13 01

Аркушів 15

2021

АНОТАЦІЯ

Документ 01116130.01197-01 13 01 «Програмний комплекс "QNX IRQ Test Toolkit". Опис програми» входить до складу програмної документації до програмного забезпечення, розробленого у результаті виконання магістерської роботи з теми «Дослідження часових характеристик затримки переривань в системах реального часу».

У даному документі представлений опис даної розробки, а саме: функціональне призначення розробки, опис логічної структури, використані технічні засоби, виклик і завантаження, вхідні і вихідні дані, опис інтерфейсу користувача, порядок роботи з програмою. Програма написана мовою C++ з використанням системних викликів операційної системи реального часу QNX. Код розроблений в інтегрованому середовищі розробки QNX Momentics IDE 4.0.1. Об'єм пам'яті, що займає програма, складає 325 Кб. Конфігурація комп'ютера стандартна. Комплекс функціонує в середовищі ОСРЧ QNX 6.3.2.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Програмний комплекс «QNX ISR Test Toolkit» складається з трьох утиліт командної строки моделювання ситуацій виникнення затримки переривань. Кожна утиліта призначена для виконання експериментальних замірів затримки обробки переривань в середовищі ОСРЧ QNX та статистичного аналізу отриманих результатів.

Для функціонування даного програмного продукту необхідно, щоб на користувацькому комп‘ютері було встановлено наступне програмне забезпечення:

- операційна система реального часу QNX Neutrino версії 6 або пізніше;
- у разі тестування на віртуальній машині — програма-гіпервізор VMware Workstation 14 або пізніше.

Програмний продукт був розроблений за допомогою мови програмування C++ та середовища розробки додатків QNX Momentics IDE версії 4.0.1.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Функціональне призначення програми полягає у виконанні багатократних замірів часу затримки обробки переривань та їх статистичної обробки, а саме підрахунку значень математичного очікування, дисперсії та стандартного відхилення.

Функціональні обмеження на програмний комплекс накладаються винятково конфігурацією ліній переривань на користувацькому комп‘ютері, яка може відрізнятись для різних машин.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1 Алгоритм програми

На рис. 3.1 представлено алгоритм роботи кожного з трьох модулів програмного комплексу у вигляді UML-діаграми діяльності.

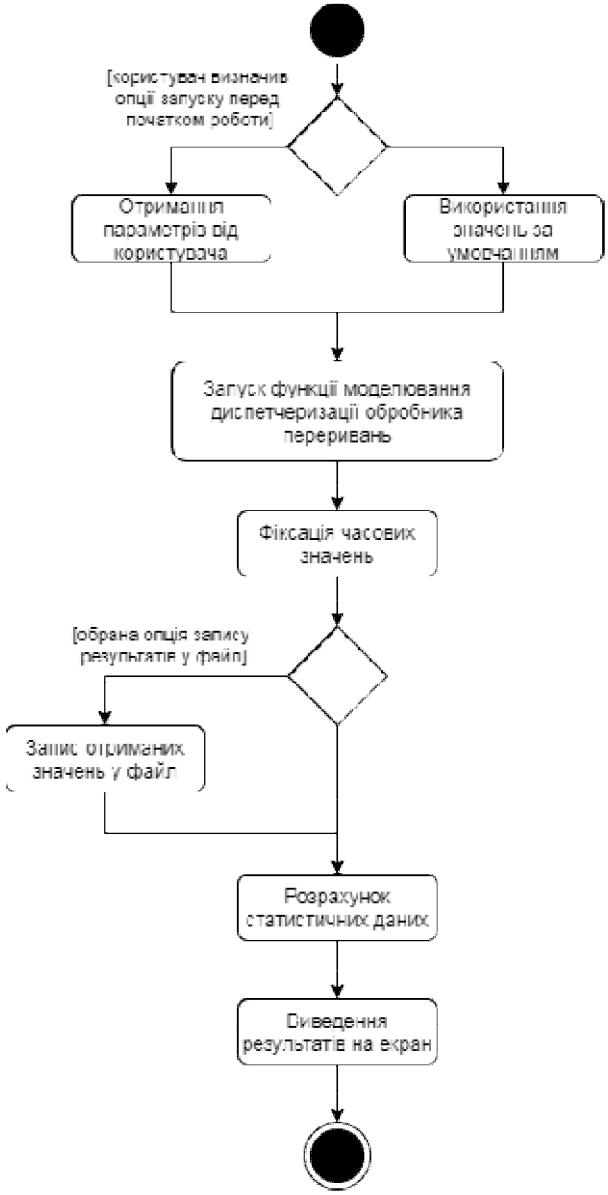

Рисунок 3.1 — Типовий алгоритм роботи програмних модулів

Далі представлені блок-схеми алгоритмів обробки переривань для кожного з модулів програмного комплексу, що розробляється.

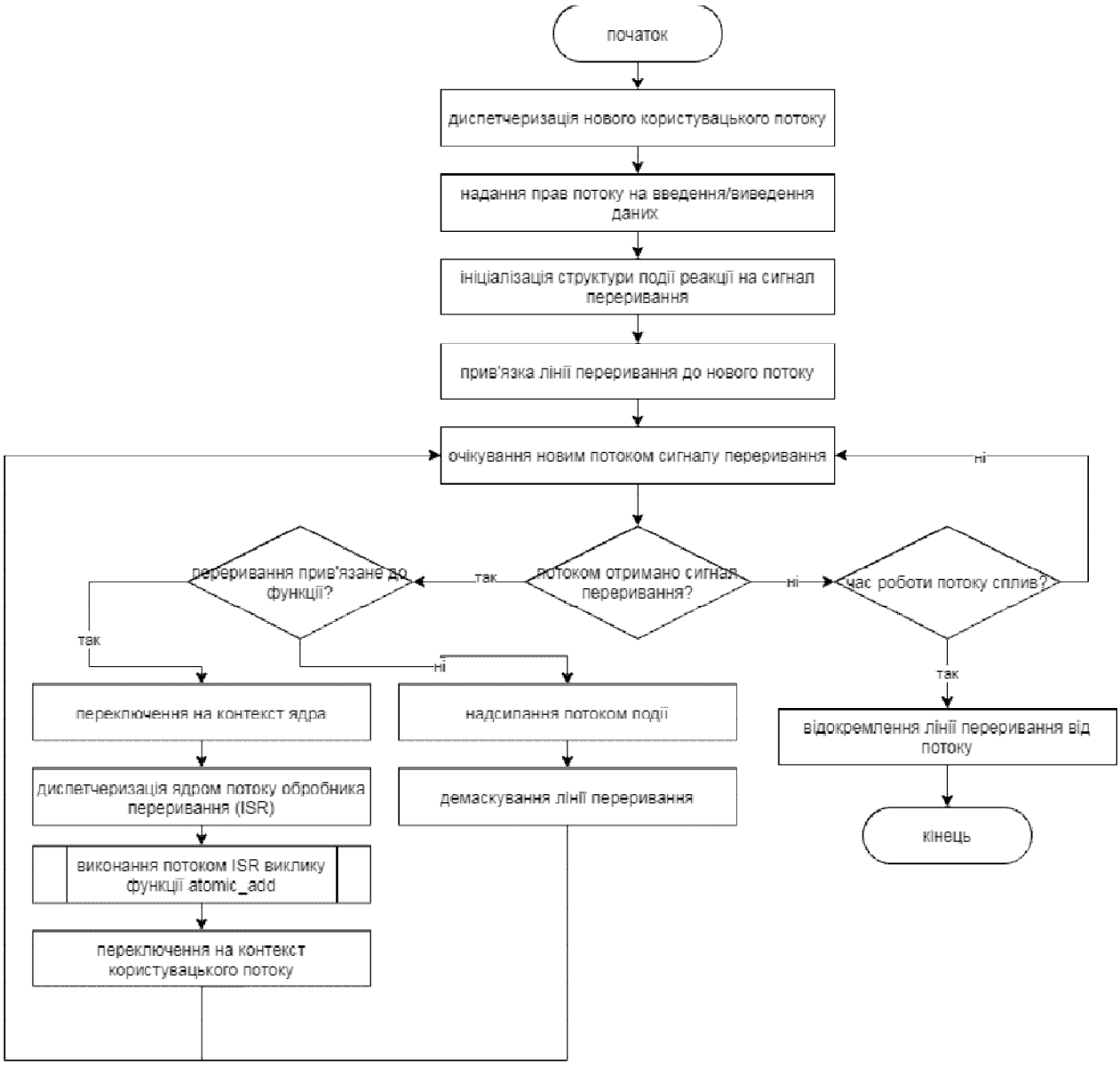

Рисунок 3.2 — Алгоритм роботи модулю затримки диспетчеризації dispatch_test

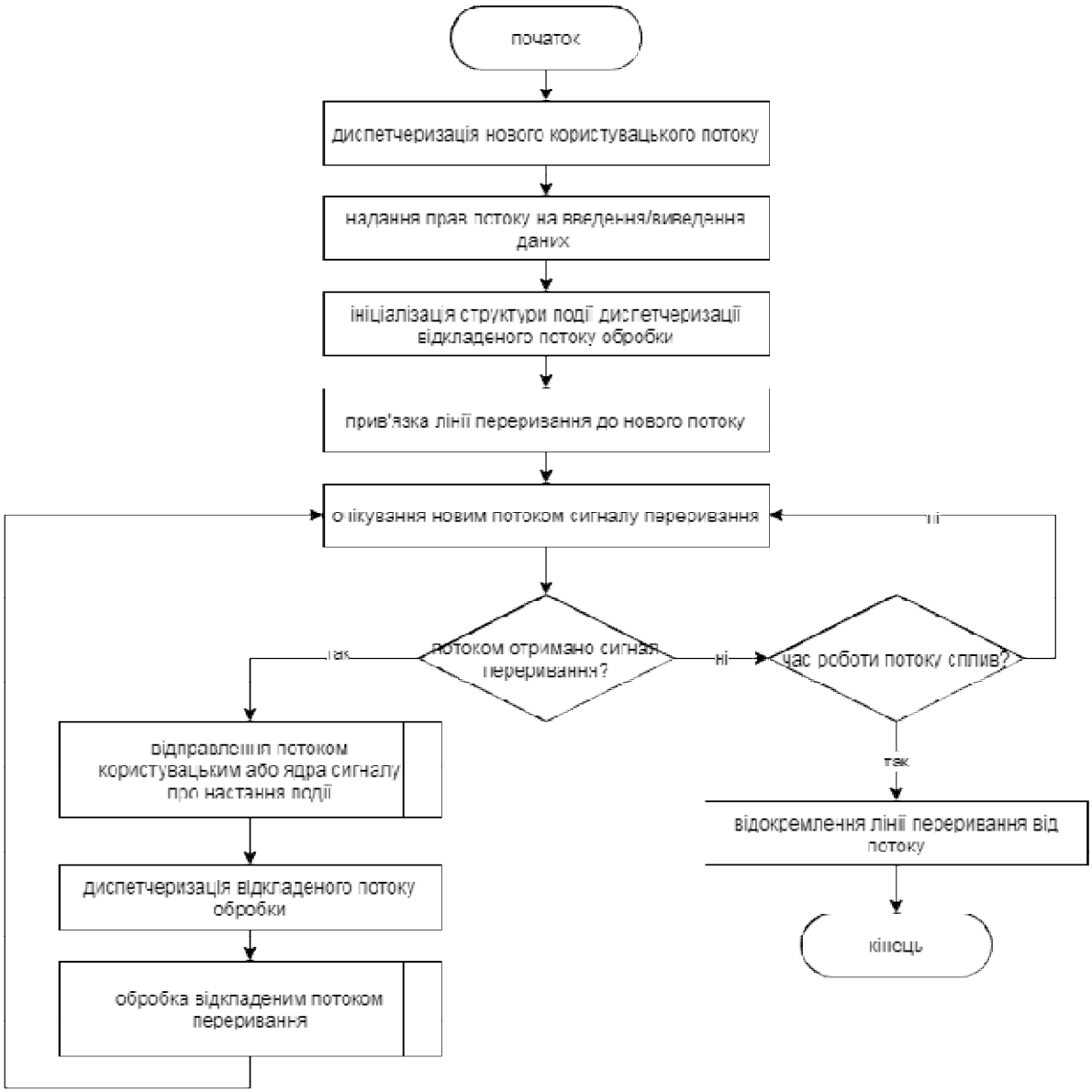

Рисунок 3.3 — Алгоритм роботи модулю затримки відкладеної обробки переривань defer_test

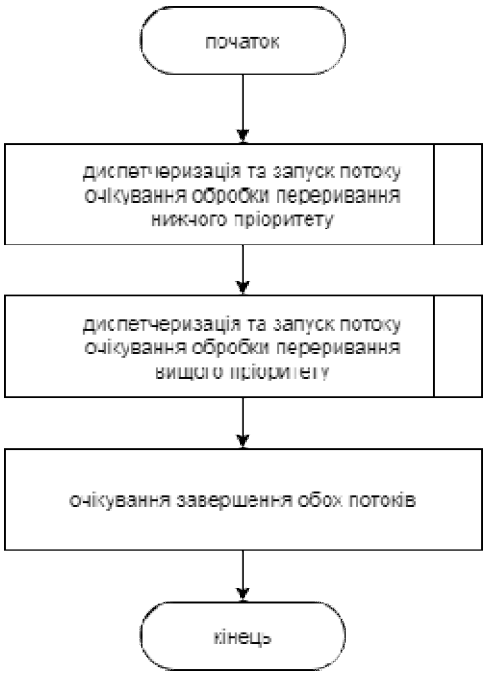

Рисунок 3.4 — Алгоритм роботи модулю затримки обробки переривань за пріоритетністю priority_test

3.2. Використані методи

Розробка даного програмного комплексу відбувається з використанням стандартних функцій та системних викликів мови C++, включаючи функції UNIX-подібних систем та функції бібліотеки стандартних системних викликів ОСРЧ QNX.

3.3. Структура програмного комплексу

На рисунку 2 відображені основні програмні модулі у якості діаграми артефактів.

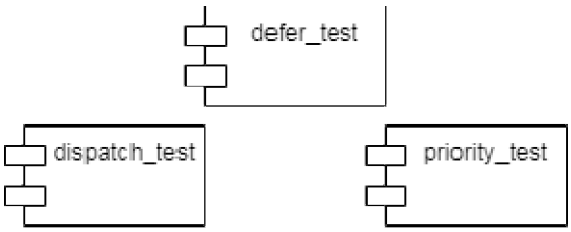

Рисунок 3.5 — Модулі програмного комплексу

3.4. Зв‘язки програмного комплексу з іншими програмами

Програмний комплекс функціонує у термінальній консолі bash операційної системи QNX, через яку, окрім виклику потрібного програмного модулю, також здійснюється введення необхідних параметрів налаштування замірів часових характеристик затримки обробки переривань.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для розробки програмного забезпечення використовується персональний комп‘ютер, що має наступні характеристики:

- процесор — мінімум одноядерний 32-, 64- або 86-бітний процесор архітектури x86;
- оперативна пам’ять — мінімум 256 МБ;
- вільна пам’ять на жорсткому диску — мінімум 350 КБ;
- периферійні пристрої — VGA-монітор з мінімальним розширенням екрану 800x600, клавіатура, миша, дисковод компакт-дисків.

5 ВИКЛИК ТА ЗАВАНТАЖЕННЯ

Перед початком роботи користувач повинен впевнитися у тому, що виконавчі файли даного програмного комплексу знаходяться на користувацькому комп‘ютері та їх можливо використовувати.

Для початку роботи з програмним комплексом користувач повинен запустити командну строку та ввести після крапки та скісної риски назву модулю для запуску, а також параметри та їх необхідні значення налаштування роботи модулю за вказівками нижче.

```
ISR priority-based handler dispatch latency measurement utility for QNX Neutrino, v1.0

Statistics (normal run):
Expected value:  0.682390858482
Variance:  0.106512506877
Standard deviation:  0.326362539021
Minimum value:  0.473190741283
Maximum value:  1.45443891005

Statistics (parallel run):
Expected value:  0.266223466058
Variance:  0.0788698329249
Standard deviation:  0.280837734154
Minimum value:  0.0796952827425
Maximum value:  1.45443891005

Results (in us):
norm prio               high prio               lower prio
1.45443891005           0.0846762379139         0.0896571930853
0.562847934369          0.0846762379139         0.0896571930853
0.747143275711          0.0647524172282         0.0946381482567
0.513038382655          0.0846762379139         0.0996191034281
0.537943158512          0.0547905068854         0.0996191034281
0.473190741283          0.0996191034281         0.0846762379139
0.488133606798          0.0846762379139         0.0946381482567
```

Рисунок 5.1 — Приклад роботи з модулем priority_test

```
ISR dispatch latency measurement utility for QNX Neutrino, v1.0

Statistics:
Expected value:  995.393235921
Variance:  7.34199508966
Standard deviation:  2.70961161233
Minimum value:  943.034280691
Maximum value:  1008.17023147

Results:
cycles          us
3356310         983.38997949
3218576         943.034280691
3368822         987.055962496
3382252         990.990917082
3381283         990.707002637
3382592         991.090536185
3396447         995.15001465
3388627         992.858775271
3397654         995.503662467
3404981         997.650454146
3372426         988.111924993
```

Рисунок 5.2 — Приклад роботи з модулем dispatch_test

6 ВХІДНІ ТА ВИХІДНІ ДАНІ

Під даними у даному випадку розуміються символьні та бінарні значення, які можуть як подаватися користувачем до програми для отримання певних результатів, так і програмою до користувача у якості відповіді на його запит.

Вхідні дані у даному випадку для кожного розробленого програмного модулю є параметри налаштування експериментальних замірів. Перед буквою параметру в обов‘язковому порядку ставиться знак «-». Числові значення деяких параметрів, що дозволяють їх встановлювати, вводяться одразу після букви параметру, з пробілом або без нього. Нижче наведено список параметрів, що можуть прийматися програмними модулями:

- [-i] — номер лінії переривання, що тестується, може бути будь-яким цілим числом;
- [-e] — опція використання виклику InterruptAttachEvent() прив‘язування події до сигналу переривання замість InterruptAttach() прив‘язки функції, параметр не приймає ніяких додаткових значень (відсутня для модулів defer_test та priority_test);
- [-f] — опція запису результатів багатократних замірів часу у файл з ім‘ям, що співпадає з номером лінії переривання, та розширенням .log , параметр не приймає ніяких додаткових значень;
- [-d] — параметр визначення тривалості проведення експериментальних замірів часу у секундах, може приймати будь-яке ціле число;
- [-w] — параметр визначення затримки початку проведення експериментальних замірів часу у секундах, може приймати будь-яке ціле число.

Вихідні дані програми:

- текстові дані щодо отриманих статистичних та експериментальних результатів з виводом на екран командної строки
- якщо була обрана опція запису результатів експериментів у файл — текстовий файл зі збереженими часовими замірами *.log, де * — номер лінії переривання, що досліджується.

7 ПОРЯДОК РОБОТИ З ПРОГРАМНИМ КОМПЛЕКСОМ

Перед початком роботи користувач повинен перевірити стан ЕОМ для запуску програм, а також дотримання вимог безпеки охорони праці та санітарних вимог на робочому місці. Використання програмного комплексу можливе лише у разі справності обладнання та дотримання вищевказаних вимог.

Усі параметри налаштування експериментів повинні вводитись латинськими символами та цифрами за допомогою клавіатури через вікно командної строки.

Порядок роботи системного адміністратора, який є відповідальним за перевірку та схвалення користувацьких дій, а також за надання консультацій щодо користування програмою, визначається окремо відповідно до розпорядження установи, у якій планується використання даного програмного продукту. Приклад порядку роботи системного адміністратора може бути наступним:

- технічний огляд серверу перед використанням — 7:45 - 8:00;
- складання плану проведення досліджень з використанням програмного комплексу, перевірка дотримання персоналом плану та умов проведення дослідження — 8:00 - 16:00;
- складання звітності за результатами досліджень та складання плану роботи на наступний день — 16:00 - 16:30;
- передача звітності та узгодження плану роботи з керівництвом — 16:45.

8 ПОВІДОМЛЕННЯ

В табл. 8.1 приведені повідомлення програми, які можуть виникати під час роботи з програмою, з позначенням, для кого призначене повідомлення, коли воно виникає та які дії необхідно виконати для продовження роботи.

Таблиця 8.1 — Опис програмних повідомлень

Текст повідомлення	Призначення	Опис ситуації	Рекомендовані дії
Unknown command detected, program terminated	Користувачеві	Користувачем було введено параметр, якого не існує	Переглянути правильність вводу параметрів командної строки перед запуском програми
Non-numeric parameter value, program terminated	Користувачеві	Користувачем було введено значення параметру, яке не є числом	Переглянути правильність вводу значень параметрів командної строки перед запуском програми
ThreadCtl returned an error	Користувачеві	Помилка функції надання прав робочому потокові на введення/виведення	Спробувати виконати програму ще раз або перезавантажити систему
InterruptAttach / InterruptAttachEvent returned an error	Користувачеві		За можливості перевірити стан лінії переривання, що перевіряється, та/або перезавантажити систему

9 БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. тра-нсп. ім. акад. В. Лазаряна, 2009. - 38 с.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

0111630. 01197-01 12 01

ПРОГРАМНИЙ КОМПЛЕКС «QNX ISR TEST TOOLKIT»

Текст програми

01116130. 01197-01 12 01

Аркушів 17

2021

АНОТАЦІЯ

Документ 01116130.01197-01 12 01 «Програмний комплекс "QNX ISR Test Toolkit". Текст програми» входить до складу програмної документації до програмного забезпечення, розробленого у результаті виконання магістерської роботи з теми «Дослідження часових характеристик затримки переривань в системах реального часу».

У даному документі представлений текст розробленої програми. Програма написана мовою C++ з використанням системних викликів операційної системи реального часу QNX. Код розроблений в інтегрованому середовищі розробки QNX Momentics IDE 4.0.1. Об'єм пам'яті, що займає програма, складає 325 Кб. Конфігурація комп'ютера стандартна. Комплекс функціонує в середовищі ОСРЧ QNX 6.3.2.

ЗМІСТ

1 СХЕМА ВЗАЄМОДІЇ МОДУЛІВ

На рис. 1.1 приведена схема взаємодії програмних модулів.

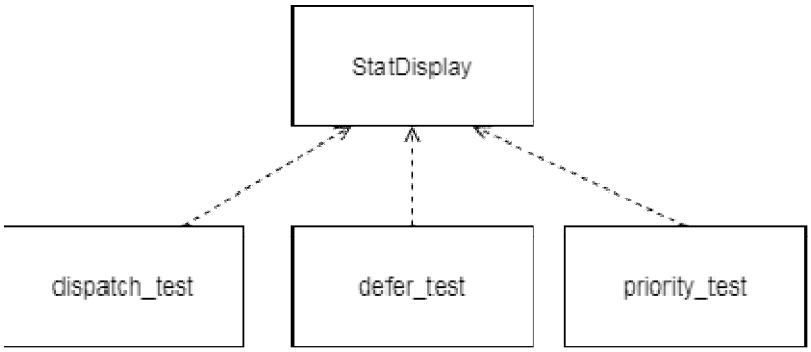

Рисунок 1.1 — Схема взаємодії модулів програмного комплексу

2 ТЕКСТ ПРОГРАМИ

2.1 Клас «StatDisplay»

2.1.1 Текст файлу-заголовку класу StatDisplay.h

```
/*Клас StatDisplay призначений для розрахунку статистичних показників для набору експериментальних значень, що міститься в екземплярі контейнеру типу std::vector*/
//Виконала: студентка групи 961м Волкодавець А.О.
#include <vector>
#include <math.h>

#ifndef STATDISPLAY_H_
#define STATDISPLAY_H_

class StatDisplay
{
private:
    int i;
    double sum, mean, sq_diff, varnc, sigma, min_value, max_value;
public:
    StatDisplay();
    void GetStats(std::vector<double>);
    double GetExpValue();
    double GetVariance();
    double GetStdDevtn();
    double GetMinValue();
    double GetMaxValue();
    virtual ~StatDisplay();
};

#endif /*STATDISPLAY_H_*/
```

2.1.2 Текст файлу StatDisplay.cc

```
#include "StatDisplay.h"
//конструктор класу
StatDisplay::StatDisplay()
{
    sum = mean = sq_diff = varnc = sigma = min_value = max_value = 0;
}
//метод отримання усіх статистичних даних
//вхідний параметр results — вектор, що містить експериментальні дані
void StatDisplay::GetStats(std::vector<double> results)
{
//підрахунок математичного очікування
  for (i = 0; i < results.size(); i++)
  {
    sum += results.at(i);
  }
    mean = sum/results.size();
//дисперсія
  for (i = 0; i < results.size(); i++)
  {
    sq_diff += (results.at(i) - sum/results.size())*(results.at(i) - sum/results.size());
  }
    varnc = sq_diff/results.size();
    //стандартне відхилення
    sigma = sqrt(sq_diff/results.size());
```

```
            for (i = 0; i < results.size(); i++)
        {
            sq_diff +=
                    (results.at(i)                    -
sum/results.size())*(results.at(i) - sum/results.size());
        }
        //знаходження мінімального значення
            for (i = 0; i < results.size(); i++)
        {
            if (min_value == 0 || min_value >
results.at(i))
            {
                    min_value = results.at(i);
            }
        }
        //знаходження максимального значен-
ня
            for (i = 0; i < results.size(); i++)
        {
            if (max_value == 0 || max_value <
results.at(i))
            {
                    max_value = results.at(i);
            }
        }
    }
    //функція повернення значення матема-
тичного очікування
    double StatDisplay::GetExpValue()
    {
            return mean;
    }
    //функція повернення значення дисперсії
    double StatDisplay::GetVariance()
    {
            return varnc;
    }
    //функція повернення значення стандар-
тного відхилення
    double StatDisplay::GetStdDevtn()
    {
            return sigma;
    }
    //функція повернення мінімального зна-
чення зі значень експериментальних даних
    double StatDisplay::GetMinValue()
    {
            return min_value;
    }
    //функція повернення максимального
значення
    double StatDisplay::GetMaxValue()
    {
            return max_value;
    }
    StatDisplay::~StatDisplay()
    {
    }
```

2.2 Модуль «dispatch_test.cc»

/*Модуль dispatch_test призначений для моделювання ситуації затримки планування елементарного потоку обробки переривання для виконання часових замірів затримки та розрахунку відповідних статистичних показників*/

```
//Виконала: студентка групи 961м Волкодавець А.О.
include <cstdlib>
#include <iostream>
```

```
#include <fstream>
#include <stdlib.h>
#include <string.h>
#include <sys/neutrino.h>
#include <inttypes.h>
#include <unistd.h>
#include <sys/syspage.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <pthread.h>
#include "StatDisplay.h"//клас розрахунку статистичних даних
struct sigevent event;//структура події для передачі обробником
int irq;//номер лінії переривання для дослідження
bool file_io_flag;//прапор запису у файл
bool using_event;//прапор прив'язки події
uint64_t cycle1, cycle2;
std::vector<double> results; //вектор запису результатів замірів у мкс
std::vector<uint64_t> results_cycles; //для звичайної роботи потоку
//функція розрахунку статистичних даних
void stat_display()
{
    StatDisplay stats;
    stats.GetStats(results);
    std::cout << "\nStatistics:\n";
    std::cout << "Expected value: " << stats.GetExpValue() <<std::endl;
    std::cout << "Variance: " << stats.GetVariance() << std::endl;
    std::cout << "Standard deviation: " << stats.GetStdDevtn() << std::endl;
    std::cout << "Minimum value: " << stats.GetMinValue() << std::endl;
    std::cout << "Maximum value: " << stats.GetMaxValue() << std::endl;
}
//функція обробки переривання
//вхідні параметри: *area - ; id
//вихідні параметри: NULL
const struct sigevent *handler(void *area, int id)
{
    cycle2 = ClockCycles( ); //фіксація часу у циклах системного таймеру
    return(&event);
}
//функція потоку прив'язки переривання
//вхідний параметр: *arg - вказівник на параметри потоку
void *irq_thread(void *arg)
{
    int id;//ідентифікатор прив'язаного потоку
    std::ofstream log;//файловий потік для запису даних
    char logname[20];//ім'я файлу
    //відкриваємо файл для запису результатів, якщо встановлено відповідний прапор
    if (file_io_flag)
    {
        itoa(irq, logname, 10);
        strcat(logname, ".log");
        log.open(logname, ios::out | ios::app);
    }
    if (ThreadCtl( _NTO_TCTL_IO, NULL ) < 0) }//надання прав потоку на введення-виведення
```

```
{
        std::cout << "ThreadCtl returned an error: "<< strerror(errno);
        return EXIT_FAILURE;
}
event.sigev_notify = SIGEV_INTR;//ініціалізація події для використання обробником
//прив'язка до переривання в залежності від обраного методу:
id = using_event
        ? InterruptAttachEvent(irq, &event, 0)//події
        : InterruptAttach(irq, &handler, NULL, 0, 0);//функції handler
//цикл очікування сигналу перериванняif (id < 0 ) //перевірка на неправильну прив'язку лінії переривання
{
        std::cout << "InterruptAttach";
        if (using_event) std::cout << "Event";
        std::cout << "() returned an error: " << strerror(errno);
        return EXIT_FAILURE;
}
while (1)
{
        cycle1 = ClockCycles( );
        InterruptWait(0, NULL);//функція очікування, що спрацює одразу після отримання сигналу переривання
        if (using_event)
        {
                cycle2 = ClockCycles( );
                InterruptUnmask(irq, id);//якщо до переривання прив'язана подія, демаскуємо лінію переривання
        }
        //запис отриманого значення часу у файл
        if (file_io_flag)
        {
                log << (double)(cycle2 - cycle1)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000) << std::endl;
        }
        //запис отриманих значень часу у відповідні вектори

    results_cycles.push_back(cycle2 - cycle1);

    results.push_back((double)(cycle2 - cycle1)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000));
        }
        //закриття файлового потоку
        if (file_io_flag)
        {
                log.close();
        }
        //від'єднання переривання від потоку
        InterruptDetach(id);
    }
    //головна функція
    int main(int argc, char *argv[])
    {
        pthread_attr_t attr;//структура атрибутів потоку
        int c, j;
```

```
unsigned int seconds, delay;//значення часу роботи та очікування початку роботи відповідно
std::cout.precision(12);
//ініціалізація значень за умовчанням
irq = 0;
delay = 0;
seconds = 1;
using_event = false;
file_io_flag = false;
std::cout << "ISR dispatch latency measurement utility for QNX Neutrino, v1.0\n";
//цикл обробки параметрів командної строки для ініціалізації значень
while((c = getopt(argc, argv, "i:efd:w:")) != -1 )
{
switch(c)
{
case 'i':
irq = atoi(optarg);
break;
case 'e':
using_event = true;
break;
case 'f':
file_io_flag = true;
break;
case 'd':
seconds = atoi(optarg);
break;
case 'w':
delay = atoi(optarg);
break;
default: //виведення списку дозволених опцій, якщо було введено помилкову опцію
std::cout << "Unknown command detected, program terminated\n";
std::cout << "Usage: ./dispatch_test \n";
std::cout << "  [-i irq_number] irq line number to test\n";
std::cout << "  [-e]  attach to event instead of function\n";
std::cout << "  [-f]  write time results to file\n";
std::cout << "  [-d seconds]  test duration in seconds\n";
std::cout << "  [-w seconds]  test start delay in seconds\n";
return EXIT_FAILURE;//те ж саме, що й return -1
}
}
//ініціалізація атрибутів потоку для запуску
pthread_attr_init( &attr );
pthread_attr_setdetachstate( &attr, PTHREAD_CREATE_DETACHED );
pthread_attr_setinheritsched( &attr, PTHREAD_EXPLICIT_SCHED );
pthread_attr_setschedpolicy( &attr, SCHED_RR );//тип диспетчеризації - карусель
attr.param.sched_priority = 10;
```

```
        sleep(delay);//затримка початку роботи для надання користувачеві часу на налаштування середовища
        pthread_create(NULL, &attr, &irq_thread, NULL); //запуск потоку прив'язки переривання
        sleep(seconds);//тут функція означає тривалість роботи нового потоку
        stat_display();//виведення статистичних даних
        //виведення результатів заміру часу на екран
        std::cout << "\nResults:\n";
        std::cout << "cycles us\n";
        for (j = 0; j < results.size(); j++)
        {
            std::cout << results_cycles.at(j) << " "<< results.at(j) << std::endl;
        }
        return EXIT_SUCCESS;//те ж саме, що й return 0
}
```

2.3 Модуль «defer_test.cc»

/*Модуль defer_test призначений для моделювання ситуації затримки планування відкладеного потоку обробки переривання для виконання часових замірів затримки та розрахунку відповідних статистичних показників*/

```
//Виконала: студентка групи 961м Волкодавець А.О.
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <string.h>
#include <stdlib.h>
#include <sys/neutrino.h>
#include <inttypes.h>
#include <unistd.h>
#include <sys/syspage.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <pthread.h>
#include <sys/siginfo.h>
#include "StatDisplay.h"

volatile uint64_t cycle1, cycle2;
struct sigevent defer_event;
int irq;
int p;
int file_io_flag;
bool using_event;
std::ofstream logger;
std::vector<double> results;
std::vector<uint64_t> results_cycles;

void stat_display()
{
    StatDisplay stats;
    stats.GetStats(results);
    std::cout << "\nStatistics:\n";
    std::cout << "Expected value: " << stats.GetExpValue() <<std::endl;
    std::cout << "Variance: " << stats.GetVariance() << std::endl;
    std::cout << "Standard deviation: " << stats.GetStdDevtn() << std::endl;
```

```
            std::cout << "Minimum value: " << stats.GetMinValue() << std::endl;
            std::cout << "Maximum value: " << stats.GetMaxValue() << std::endl;
        }
        //функція відкладеної обробки переривання
        //вхідний параметр: weh - значення отриманого сигналу від потоку ядра
        void isr_thread(union sigval weh)
        {
            cycle2 = ClockCycles();
            if (file_io_flag)
            {
                logger << (double)(cycle2 - cycle1)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000) << std::endl;
            }
            results_cycles.push_back((uint64_t)cycle2 - cycle1);
            results.push_back((double)(cycle2 - cycle1)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000));
            pthread_exit(NULL);
        }
        //функція-потік ядра відкладеного планування потоку-обробки
        const sigevent *deferred_thread(void *area, int size)
        {
            cycle1 = ClockCycles();
            return (&defer_event);
        }

        void *irq_thread(void *arg)
        {
            int id;
            if (ThreadCtl( _NTO_TCTL_IO, NULL ) < 0) }//надання прав потоку на введення-виведення
            {
                std::cout << "ThreadCtl returned an error: "<< strerror(errno);
                return EXIT_FAILURE;
            }

            id = InterruptAttach(irq, &deferred_thread, NULL, 0, 0);
            if (id < 0 ) //перевірка на неправильну прив'язку лінії переривання
            {
                std::cout << "InterruptAttach() returned an error: " << strerror(errno);
                return EXIT_FAILURE;
            }
            SIGEV_THREAD_INIT(&defer_event, &isr_thread, NULL, NULL);

            while (1)
            {
                InterruptWait(0, NULL);
            }
            InterruptDetach(id);
        }

        int main(int argc, char *argv[])
        {
            pthread_attr_t attr;
            int c, j;
            unsigned int seconds, delay;
            char logname[20];
            std::cout.precision(12);
```

```
        irq = 0;
        delay = 0;
        seconds = 1;
        file_io_flag = false;
        while((c = getopt(argc, argv, "i:fd:w:")) != -1 )
        {
            switch(c)
            {
                case 'i':
                    irq = atoi(optarg);
                case 'f':
                    file_io_flag = true;
                    break;
                case 'd':
                    seconds = atoi(optarg);
                    break;
                case 'w':
                    delay = atoi(optarg);
                    break;
                default:
                    std::cout << "Unknown command detected, program terminated\n";
                    std::cout << "Usage: ./dispatch_test \n";
                    std::cout << "      [-i irq_number] irq line number to test\n";
                    std::cout << "      [-f]      write time results to file\n";
                    std::cout << "      [-d seconds]      test duration in seconds\n";
                    std::cout << "      [-w seconds]      test start delay in seconds\n";
                    break;
            }
        }
        if (file_io_flag)
        {
            itoa(irq, logname, 10);
            strcat(logname, ".log");
            logger.open(logname, ios::out | ios::app);
        }
        pthread_attr_init( &attr );
        pthread_attr_setdetachstate( &attr, PTHREAD_CREATE_DETACHED );
        pthread_attr_setinheritsched( &attr, PTHREAD_EXPLICIT_SCHED );
        pthread_attr_setschedpolicy( &attr, SCHED_RR );
        attr.param.sched_priority = 10;
        sleep(delay);
        pthread_create(NULL, &attr, &irq_thread, NULL);
        sleep(seconds);

        if (file_io_flag)
        {
            logger.close();
        }

        stat_display();

        std::cout << "\nResults:\n";
        std::cout << "cycles      us\n";
        for (j = 0; j < results.size(); j++)
        {
```

```
            std::cout
            << results_cycles.at(j) << "
        "<< results.at(j) << std::endl;
        }
        return EXIT_SUCCESS;
    }
```

2.4 Модуль «priority_test.cc»

/*Модуль priority_test призначений для моделювання ситуації затримки роботи потоку обробки переривання за умови, що паралельно виконується потік обробки переривання, вищого за пріоритетом, для виконання часових замірів тривалості роботи та розрахунку відповідних статистичних показників */

```
//Виконала: студентка групи 961м Волкодавець А.О.
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <string.h>
#include <stdlib.h>
#include <sys/neutrino.h>
#include <inttypes.h>
#include <unistd.h>
#include <sys/syspage.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <atomic.h>
#include <pthread.h>
#include "StatDisplay.h"
struct sigevent defer_event; //єкземпляр структури події для використання у обробнику переривання
volatile uint64_t cycle1, cycle2, cycle3, cycle4; //змінні для часових значень у циклах
struct sched_param param; //параметри планування потоків
volatile unsigned dummy_value; //змінна для редагування обробниками переривань
bool fancy_flag; //прапор для відмічення початку роботи нового потоку
//екземпляр структури вектору для збереження часових даних у мікросекундах
std::vector<double> time_vector; //для звичайної роботи потоку
std::vector<double> time_vector2;
std::vector<double> prio_vector; //для потоку з паралельним потоком вищого пріоритету
int irq;
bool file_io_flag;
bool using_event;
std::ofstream logger;
//функція отримання статистичних даних
void stat_display()
{
    StatDisplay stats;
    stats.GetStats(time_vector);
    std::cout << "\nStatistics (normal run):\n";
    std::cout << "Expected value: " << stats.GetExpValue() <<std::endl;
    std::cout << "Variance: " << stats.GetVariance() << std::endl;
    std::cout << "Standard deviation: " << stats.GetStdDevtn() << std::endl;
    std::cout << "Minimum value: " << stats.GetMinValue() << std::endl;
    std::cout << "Maximum value: " << stats.GetMaxValue() << std::endl;
    stats.GetStats(prio_vector);
```

```
            std::cout << "\nStatistics (parallel run):\n";
            std::cout << "Expected value: " << stats.GetExpValue() <<std::endl;
            std::cout << "Variance: " << stats.GetVariance() << std::endl;
            std::cout << "Standard deviation: " << stats.GetStdDevtn() << std::endl;
            std::cout << "Minimum value: " << stats.GetMinValue() << std::endl;
            std::cout << "Maximum value: " << stats.GetMaxValue() << std::endl;}
        //функція-обробник переривання
        const sigevent *isr_thread2(void *area, int size)
        {
            //заміри часу виконання функції
            cycle1 = ClockCycles();
            atomic_add(&dummy_value, 1); //інкрементування змінної dummy_value безпечною функцією
            cycle2 = ClockCycles();
            return (&defer_event); //обробник повертає подію для повідомлення основного потоку про своє завершення
        }
        //функція-обробник переривання без замірів часу
        const sigevent *isr_thread3(void *area, int size)
        {
            cycle3 = ClockCycles();
            atomic_sub(&dummy_value, 1);
            cycle4 = ClockCycles();
            return (&defer_event);
        }

        //функція потоку, що реагує на переривання вищого пріоритету (переривання системного таймеру)
        void *hiprio_thread(void *arg)
        {
            int id;
            if (ThreadCtl( _NTO_TCTL_IO, NULL ) < 0) }//запит на надання привілегій потоку на приєднання обробника переривань
            {
                std::cout << "ThreadCtl returned an error: "<< strerror(errno);
                return EXIT_FAILURE;
            }
            SIGEV_INTR_INIT(&defer_event); //ініціалізація екземпляру структури події для використання
            id = InterruptAttach(0, &isr_thread3, NULL, 0, 0); //приєднуємо до даного потоку обробник переривання системного таймеру (на лінії IRQ0)
            if (id < 0 ) //перевірка на неправильну прив'язку лінії переривання
            {
                std::cout << "InterruptAttach() returned an error: " << strerror(errno);
                return EXIT_FAILURE;
            }

            while (1)
            {
                InterruptWait(0, NULL);//поток циклічно очікує повідомлення про сигнал переривання

                time_vector2.push_back
```

```
((double)(cycle4 - cycle3)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000));

        }
        InterruptDetach(id);//від'єднання переривання від даного потоку
        pthread_exit(NULL);//завершення роботи потоку
    }
    //функція потоку, що реагує на переривання нижчого пріоритету (переривання контролеру жорсткого диску)
    void *loprio_thread(void *arg)
    {
        int id;
        double result;
        ThreadCtl( _NTO_TCTL_IO, 0 );//запит на привілегії прив'язування переривання
        SIGEV_INTR_INIT(&defer_event);//ініціалізація події
        id = InterruptAttach(irq, &isr_thread2, NULL, 0, 0);//прив'язка переривання та його обробника до потоку
        while (1)
        {
            InterruptWait(0, NULL);//поток циклічно очікує повідомлення про сигнал переривання
            result = (double)(cycle2 - cycle1)/(SYSPAGE_ENTRY(qtime)->cycles_per_sec/1000000);
            if (!fancy_flag)
            {
                //якщо прапор не активований (має значення брехні), потік виконується у нормальному режимі, записуємо часові дані в перший вектор
                //if (time_vector.back() != result)

                time_vector.push_back (result);
            }
            else
            {
                //якщо прапор активований (має значення істини), то паралельно виконується потік з перериванням вищого пріоритету, запис часових даних в другий вектор

                prio_vector.push_back (result);
                if (file_io_flag)
                {
                    logger << prio_vector.back() << std::endl;
                }
            }
        }
        InterruptDetach(id); //від'єдання переривання
        pthread_exit(NULL);//завершення потоку
    }
    //головна функція
    int main(int argc, char *argv[])
    {
        pthread_attr_t attr; //атрибути потоків
        int c;
        unsigned int j; //змінна-вказівник для виводу даних у векторах
```

```
unsigned int seconds, delay;
char logname[20];
std::cout.precision(12);
irq = 0;
delay = 0;
seconds = 1;
using_event = false;
file_io_flag = false;
fancy_flag = false;//встановлюємо прапор як неактивований
while((c = getopt(argc, argv, "i:efd:w:")) != -1 ) //цикл зчитування та обробки введених параметрів командної строки
{
switch(c)
{
case 'i':
irq = atoi(optarg);
break;
case 'e':
using_event = true;
break;
case 'f':
file_io_flag = true;
break;
case 'd':
seconds = atoi(optarg);
break;
case 'w':
delay = atoi(optarg);
break;
default:
std::cout << "Unknown command detected, program terminated\n";
std::cout << "Usage: ./dispatch_test \n";
std::cout << " [-i irq_number] irq line number to test\n";
std::cout << " [-e] attach to event instead of function\n";
std::cout << " [-f] write time results to file\n";
std::cout << " [-d seconds] test duration in seconds\n";
std::cout << " [-w seconds] test start delay in seconds\n";
break;
}
}
sleep(delay); //процес призупиняє роботу для налаштування оператором середовища дослідження
pthread_t lowprio, highprio;//змінні на позначення потоків
//ініціалізація структури атрибутів створення потоку
pthread_attr_init( &attr );
pthread_attr_setdetachstate( &attr, PTHREAD_CREATE_DETACHED );
pthread_attr_setinheritsched( &attr, PTHREAD_EXPLICIT_SCHED );//політика диспетчеризації
pthread_attr_setschedpolicy( &attr, SCHED_FIFO); //тип диспетчеризації
attr.param.sched_priority = 10;//номер пріоритету
```

```
pthread_create(&lowprio, &attr, &loprio_thread, NULL); //запуск потоку відстеження переривання нижчого пріоритету
sleep(seconds);
fancy_flag = true;//прапор активовано
pthread_create(&highprio, &attr, &hiprio_thread, NULL);//запуск потоку відстеження переривання вищого пріоритету
sleep(3);//обидва потоки виконуються паралельно
//цикл виводу отриманих часових значень з векторів для попарного порівняння

if (file_io_flag)
{
    itoa(irq, logname, 10);
    strcat(logname, ".log");
    logger.open(logname, ios::out | ios::app);
}

if (file_io_flag)
{
    logger.close();
}
stat_display();
//виведення результатів часових замірів на екран
std::cout << "\nResults (in us):\n";
std::cout << "norm prio high prio lower prio\n";
for (j = 0; j < prio_vector.size(); j++)
{
    std::cout << time_vector.at(j) << "
"<< time_vector2.at(j) << "
" << prio_vector.at(j) << std::endl;
}
return EXIT_SUCCESS;
}
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

0111630. 01197-01 ІЗ 01

ПРОГРАМНИЙ КОМПЛЕКС «QNX ISR TEST TOOLKIT»

Керівництво користувача. Керівництво з використання програмного комплексу «QNX ISR Test Toolkit»

01116130. 01197-01 ІЗ 01

Аркушів 11

2021

АНОТАЦІЯ

Документ 01116130.01197-01 ІЗ 01 «Програмний комплекс "QNX ISR Test Toolkit". Керівництво користувача. Керівництво з використання програмного комплексу "QNX ISR Test Toolkit"» входить до складу програмної документації до програмного забезпечення, розробленого у результаті виконання магістерської роботи з теми «Дослідження часових характеристик затримки переривань в системах реального часу».

У даному документі представляється керівництво з використання розробленої програми, що включає дані про призначення програми, файловий склад дистрибутивного носія, інструкції щодо підготовки та безпосередньо роботи з програмою, опис операцій . Програма написана мовою С++ з використанням системних викликів операційної системи реального часу QNX. Код розроблений в інтегрованому середовищі розробки QNX Momentics IDE 4.0.1. Об'єм пам'яті, що займає програма, складає 325 Кб. Конфігурація комп'ютера стандартна. Комплекс функціонує в середовищі ОСРЧ QNX 6.3.2.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Програмний комплекс являє собою набір утиліт, що призначені для виконання експериментальних замірів затримки обробки переривань з автоматичними фіксацією та статистичним аналізом отриманих часових характеристик.

Список утиліт комплексу є наступним:

- dispatch_test — модуль тестування часових значень затримки звичайного планування обробки переривань;
- defer_test — модуль тестування часових значень затримки відкладеної обробки переривань;
- priority_test — модуль тестування часових значень затримки обробки переривань нижчого пріоритету.

2 ВИМОГИ ДО КОРИСТУВАННЯ

2.1 Системні вимоги

Програма призначена для функціонування в середовищі з наступними характеристиками:

- процесор — мінімум одноядерний 32-, 64- або 86-бітний процесор архітектури x86;
- оперативна пам’ять — мінімум 256 МБ;
- вільна пам’ять на жорсткому диску — мінімум 350 КБ;
- периферійні пристрої — VGA-монітор з мінімальним розширенням екрану 800x600, клавіатура, миша, дисковод компакт-дисків.

Також на робочій ЕОМ повнні бути встановлені наступні програмні компоненти:

- операційна система реального часу QNX Neutrino версії 6 або пізніше;
- програми-драйвери периферійних пристроїв, переривання яких будуть тестуватися даною програмою.
- у разі тестування на віртуальній машині — програма-гіпервізор VMware Workstation 14 або пізніше.

2.1 Необхідні навички

Для роботи з програмним комплексом від користувача вимагається наявність таких знань, вмінь та навичок:

- базові навички користування персональним комп‘ютером та периферійними пристроями (клавіатурою та мишею);
- навички роботи з командною строкою Unix-подібних ОС, знання основних команд Unix та вміння їх застосовувати;
- навички роботи з операційною системою реального часу QNX та базове розуміння принципів її роботи;
- у разі, якщо використовується версія QNX з вбудованим графічним інтерфейсом — базові навички користування графічним інтерфейсом Photon;

- базові вміння роботи з пристроями генерування системних переривань, що досліджуються — підключення, налаштування, перевірка коректної роботи;
- розуміння таких понять математичної статистики, як математичне очікування та стандартне відхилення, та вміння застосувати їх на практиці.

3 ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

Даний програмний комплекс є портативним, тобто, відсутній етап запуску утиліти встановлення даної програми на комп‘ютер. Для роботи з програмою достатньо лише запустити необхідний виконавчий файл з будь-якої директорії.

Для початку роботи з програмами запустіть термінал командної строки та введіть після крапки та скісної риски назву модулю для запуску, а також параметри та їх необхідні значення налаштування роботи модулю за вказівками, зазначеними нижче.

Усі файли, що створюються програмою, зберігаються в директорії, у якій розташований виконавчий файл програми.

3.1 Порядок проведення експерименту

Перед запуском експериментальних замірів налаштуйте відповідні параметри на позначення експериментальних умов:

- [-i] — номер лінії переривання, що тестується, може бути будь-яким цілим числом;
- [-e] — опція використання виклику InterruptAttachEvent() прив‘язування події до сигналу переривання замість InterruptAttach() прив‘язки функції, параметр не приймає ніяких додаткових значень. Цей параметр доступний лише для модуля дослідження затримки диспетчеризації dispatch_test;
- [-f] — опція запису результатів багатократних замірів часу у файл з ім‘ям, що співпадає з номером лінії переривання, та розширенням .log , параметр не приймає ніяких додаткових значень;
- [-d] — параметр визначення тривалості проведення експериментальних замірів часу у секундах, може приймати будь-яке ціле число;
- [-w] — параметр визначення затримки початку проведення експериментальних замірів часу у секундах, може приймати будь-яке ціле число.

Букви параметрів вводяться після знаку «-» та розділяються між собою одним пробілом. Пробіл між буквою параметру та його числовим значенням ставити не обов‘язково.

3.2 Обробка експериментальних даних

Після того, як експеримент було завершено, на вікні терміналу з‘являться наступні елементи:

1. блок розрахованих статистичних даних — включає значення математичної статистики, дисперсії, стандартного відхилення, мінімального та максимального значень;
2. блок отриманих замірів часу — в залежності від обраного модулю може містити наступні дані
    a. для модулів dispatch_test та defer_test — значення затримки в циклах системного таймеру та мкс;
    б. для модулю priority_test — значення затримки в мкс для потоку обробки переривання низького пріоритету за самостійного виконання, паралельного потоку високого пріоритету та потоку низького пріоритету за паралельного виконання з потоком більш високого пріоритету.

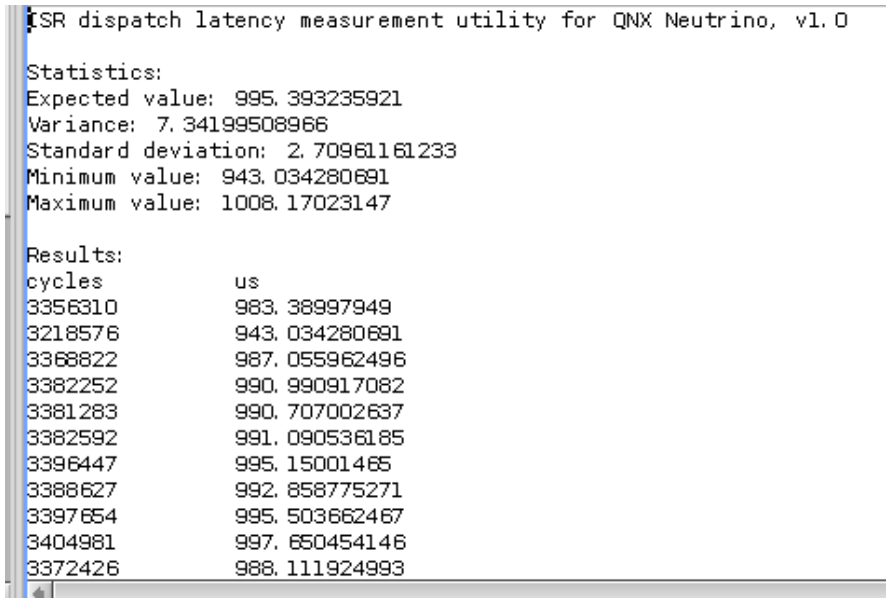

Рисунок 3.1 — Приклад результатів роботи для модулю dispatch_test

```
ISR deferred handler dispatch latency measurement utility for QNX Neutrino, v1.0

Statistics:
Expected value:  100.577496898
Variance:  773.242330295
Standard deviation:  27.8072352149
Minimum value:  24.237327864
Maximum value:  956.921183709

Results:
cycles          us
367625          107.713155582
308941          90.5188983299
345083          101.108409024
379984          111.334309991
360706          105.685906827
309570          90.7031936713
347106          101.70114269
366554          107.399355406
367064          107.548784061
310199          90.8874890126
350812          102.786990917
```

Рисунок 3.2 — Приклад результатів роботи для модулю defer_test

```
ISR priority-based handler dispatch latency measurement utility for QNX Neutrino, v1.0

Statistics (normal run):
Expected value:  0.682390858482
Variance:  0.106512506877
Standard deviation:  0.326362539021
Minimum value:  0.473190741283
Maximum value:  1.45443891005

Statistics (parallel run):
Expected value:  0.266223466058
Variance:  0.0788698329249
Standard deviation:  0.280837734154
Minimum value:  0.0796852827425
Maximum value:  1.45443891005

Results (in us):
norm prio               high prio               lower prio
1.45443891005           0.0846762379139         0.0896571930853
0.562847934369          0.0846762379139         0.0896571930853
0.747143275711          0.0647524172282         0.0946381482567
0.513038382655          0.0846762379139         0.0996191034281
0.537943158512          0.0547905068854         0.0996191034281
0.473190741283          0.0996191034281         0.0846762379139
0.488133606798          0.0846762379139         0.0946381482567
```

Рисунок 3.3 — Приклад результатів роботи для модулю priority_test

Якщо була обрана опція запису результатів часових замірів у файл, у папці з виконавчим файлом обраного модулю з‘явиться файл *.log, де * — номер лінії переривання, що досліджується. При повторному запису у файл результатів до вже існуючих даних файлу додаються нові дані.

Для повторного проведення експерименту знову виконайте процедуру, описану у п.п. 3.1.

4 РЕКОМЕНДАЦІЇ ЩОДО ЗАСТОСУВАННЯ

Нижче наведені деякі рекомендації щодо більш безпечного та продуктивного використання програмного комплексу.

На початку роботи для отримання коректних та очікуваних результатів необхідно переконатися у підключенні та справному функціонуванні усіх пристроїв, а також знати, які досліджувані пристрої займають ту чи іншу лінію переривання. Для перевірки належності пристроїв до ліній переривання використовується утиліта командної строки pci -v.

Перед першим використанням програм рекомендується провести декілька їх тестових запусків, зокрема, з параметрами по умовчанню та з параметрами, необхідними для дослідження, задля ознайомлення з ходом використання програми.

Рекомендується якомога частіше робити резервні копії даних, отриманих під час виконання програми, для уникнення їх втрати за раптового порушення роботи комп‘ютеру.

Для ефективного використання програм за умов наявності відповідних знань та навичок рекомендується написання bash-скриптів почергового запуску утиліт з обраними параметрами. Однак потрібно зауважити, що не всі параметри є сумісними з усіма програмами, тому перед використанням зверніться до розділу 3 даного керівництва користувача за інформацією щодо сумісних опцій для кожної утиліти.

Тривале використання програми може призвести до перенапруження зору та мозкової активності, тому рекомендується через кожні 25 хвилин користування програмою робити п‘ятихвилинні перерви поза робочим місцем користувача. Під час перерви доцільно виконувати легкі фізичні вправи та зорову гімнастику. За можливості потрібно також забезпечити мінімізацію випромінювання екрану за допомогою встановлення спеціального захисного екрану на монітор або використання спеціальних захисних окулярів під час роботи.

5 ЗАСТЕРЕЖЕННЯ ЩОДО ЗАСТОСУВАННЯ

5.1 Заходи безпеки під час користування

Організація робочого місця з комп‘ютером, на якому знаходиться програмний комплекс, не повинна завдавати перешкод або дискомфорту під час використання програм. Для цього такі компоненти, як клавіатура та миша, повинні мати зручне для використання положення на робочому столі та достатню довжину кабелю для підключення до системного блоку для зміни положення у будь-який зручний момент. Кабелі підключення апаратних компонентів не повинні сплутуватись, оскільки крім незручного їх використання та переміщення є також небезпека їх пошкодження.

Забороняється використання програми при наявних пошкодженнях та/або задимлення апаратних складових комп‘ютеру або нестабільній подачі електроенергії.

Не допускається використання програмного забезпечення за некоректної установки операційної системи QNX на робочий комп‘ютер, що супроводжується збоями у роботі ядра або основних процесів та/або збоями у фізичній або віртуальній пам‘яті.

5.2 Аварійні ситуації

Якщо під час використання програми виникають програмні збої, викликані збоями у програмній пам‘яті або інших програмних компонентах операційної системи QNX, потрібно перезапустити систему.

Якщо збої у функціонуванні відбулись через несанкціоноване втручання у дані третіми особами або через підозрілі програмні процеси, потрібно негайно сповістити про це системного адміністратора.

Якщо під час використання програми виникають збої у роботі апаратних компонентів, переривання яких досліджуються, потрібно передчасно завершити виконання програми, вимкнути комп‘ютер та перевірити стан пристроїв.

Тези до XIV Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (15 – 16 грудня 2021 року, м. Дніпро), секція «Інтелектуальні інформаційні та телекомунікаційні технології промислових і транспортних систем» (неопубліковані).

ДОСЛІДЖЕННЯ ВПЛИВУ ЗАТРИМОК ОБРОБКИ ПЕРЕРИВАНЬ В ОПЕРАЦІЙНИХ СИСТЕМАХ РЕАЛЬНОГО ЧАСУНА WCET

Автор: Волкодавець А. О., студентка групи ПЗ2021
Науковий керівник: кандидат технічних наук, доцент Нечай В. Я.
Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Під операційною системою реального часу (ОСРВ) розуміється така система, що призначена для додатків, що виконуються у реальному часі та потребують негайної реакції на зміни в програмному середовищі. Для забезпечення надійності та коректності роботи системи під час розв'язку таких задач дуже важливим є питання аналізу та прогнозування апріорі максимального часу, що витрачається на розв'язок - метрики Worst-Case ExecutionTime (WCET). Метою даної дослідницької роботи є аналіз та оцінка вкладу в значення WCET системних переривань та затримки їх обробки на прикладі операційної системи реального часу QNX Neutrino.

В ОСРВ QNX обробка переривань виконується у два кроки - отримання сигналу про виникнення переривання та безпосередньо обробка переривань за допомогою стандартних обробників або користувацьких функцій обробки. Для операційної системи QNX можна виділити наступні три ситуації, які можуть затримувати обробку переривань:

1. Затримка черги. Виконання потоків обробки переривання може гальмуватись за недостачі вільних векторів переривання, завантаженості системи у конкретний момент часу, особливостей реалізації конкретної апаратної архітектури або через складність коду функції-обробника.
2. Витіснення потоку обробки більш пріоритетними задачами. Диспетчеризація потоків, що передбачена в QNX, ставить потоки до черги виконання, базуючись на значенні їх пріоритетів, тому є можливим витіснення потоків-обробників іншими потоками, що мають більшу пріоритетність. До того ж, існує така розповсюджена проблема, як інверсія пріоритетів - момент, коли за пріоритетністю виконується не той потік, якому необхідно працювати в конкретний момент часу.
3. Відкладені переривання. Існує можливість виникнення ситуації, при якій системою був отриманий сигнал від обробника переривання високого пріоритету, однак сама обробка переривання за певних обставин спрацьовує не одразу.

У даній дослідницькій роботі планується окремо розглянути кожну з презентованих ситуацій на предмет оцінки їх впливу на час обробки переривань, а також дослідити необхідність у використанні власних функцій обробки переривань для дослідження часових характеристик їх затримки.

Актуальність результатів даного дослідження полягає в оцінці вкладу затримки переривань у WCET.

Тези до Всеукраїнської науково-технічної конференції молодих учених, магістрантів та студентів «Науково-технічний прогрес на транспорті» (29 березня 2021 року).

A.O. Volkodavets'
Research supervisor: V. Y. Nechay, Candidate of Technical Sciences, Professor
Language supervisor: A. O. Muntyan, Candidate of Philological Sciences, Associate Professor

Dnipro National University of Railway Transportnamed after Academician V. Lazarian.

MEASURING THE IMPACT OF INTERRUPT DELAYS IN REAL-TIME OPERATING SYSTEMS

Real time operating systems (RTOS) are designed for tasks that require immediate responses to any changes in the environment, or hard real-time tasks. Hardware and software interrupts play a significant part in ensuring that the system runs and performs all the tasks correctly and promptly, hence it is important to be able to analyze and predict the time of task execution while bearing in mind the usage of interrupt calls and their respective handler functions, as well as specific RTOS architecture details, such as task scheduling, priorities etc.

The possible situations that may lead to interrupt delays are as follows:

1. Queue delays – multi-core system architecture, complex interrupt handlers and such other factors influence the efficiency of the system during task queue processing.
2. Priority issues – inversion of task priority is a common problem in RTOSs that needs to be handled carefully when programming and running multi-threaded applications.
3. Postponed interrupts – sometimes the system can postpone the interrupt handling upon receiving its interrupt signal. This also needs to be taken into consideration when planning and writing the interrupt handler portion of the code.

The goal of this research project is to examine the impact of the presented interrupt delay situations on the time of handling hardware and software interrupts. For this, a metric known as Worst Case Execution Time (WCET) will be used. WCET describes the longest time interval a task can be executed for. This metric provides crucial insight into the way real-time operating systems execute hard real-time tasks. There are three groups of methods of WCET evaluation – analytical (static), measurement-based (dynamic) and hybrid methods.

Static measurement approach allows application analysis based on its source code structure and the methods and system calls used. The main drawback of using static methods lies in the necessity of complex hardware and software for precise diagnosis.

Measurement-based methods are used in absence of the source code of the application. These techniques mostly consist of running benchmarks on the program. Dynamic analysis cannot guarantee the safety of the so-called "raw" WCET value obtained by its means, which is why it is not suitable for real-time systems.

Unfortunately, a great number of experiments conducted on measuring WCET for RTOS-specific tasks shows that there is no universal way of providing fast and accurate results with limiting oneself to using either static or dynamic methods only. That is where the hybrid approach comes in. Hybrid methods combine various positive aspects of static and dynamic techniques for the optimal calculation speed and measurement results. New hybrid methods are constantly being developed and researched, albeit mainly for specific situations that are not restricted to RTOS usage, which is why hybrid methods can still be inefficient unless they are tailored to the task at hand.

The practical aim for this research project is not only to determine an effective way for WCET measurement and the impact of interrupt delays on the overall WCET value in RTOSs, but also to highlight the tasks in various scientific fields for which the longevity of interrupt delays is vital for successful execution.

Тези до 81 Всеукраїнської науково-технічної конференції молодих учених, магістрантів та студентів «Наука і сталий розвиток транспорту» (28 жовтня 2021 року) (неопубліковані)

ОСОБЛИВОСТІ ОБРОБКИ ПЕРЕРИВАНЬ В ОПЕРАЦІЙНІЙ СИСТЕМІ РЕАЛЬНОГО ЧАСУ QNX ТА ЕКСПЕРИМЕНТАЛЬНОГО ЗАМІРУ ЧАСУ ЗАТРИМКИ ЇХ ОБРОБКИ

Автор: Волкодавець А. О., студентка групи ПЗ2021
Науковий керівник: кандидат технічних наук, доцент Нечай В. Я.
Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Операційна система реального часу (ОСРВ) QNX Neutrino використовується у програмних системах, які націлені на розв‘язок певних задач реального часу. Переривання є одним з важливих чинників впливу на параметр Worst Case Execution Time (WCET), тобто, найгірший час виконання програм, оскільки вони є засобами комунікації системи з підключеними пристроями та внутрішніми компонентами, такими, як таймер реального часу та математичний співпроцесор. Оскільки успішний розв‘язок задач реального часу повністю залежить від вчасної реакції системи, важливо мати дані апріорі щодо можливих затримок у роботі програм, що містять функції обробки переривань.

Більшість існуючих посібників з дослідження часових характеристик обробки переривань містять рекомендації щодо виконання вимірів на апаратному рівні. У даній дослідницькій роботі, направленій на дослідження цих часових характеристик, переривання розглядаються саме з програмної точки зору, так як програмні засоби ОСРВ QNX є універсальними у використанні для всіх існуючих архітектур та моделей процесорів.

Під час дослідження часу між моментом отримання сигналу на відповідній лінії переривання та початку виконання функції обробки цього сигналу в ОСРВ QNX було відмічено наступне:

1. Затримка між безперервно виникаючими перериваннями на одній й тій самій лінії відбувається за майже однакові проміжки часу. Це можливо помітити, наприклад, під час дослідження переривань системного таймеру. За отриманими даними експериментальних замірів різниця між часовими значеннями затримки таких переривань не перевищує 1 мс.
2. В залежності від обраного типу переривань фіксується декілька значень їх затримки. Наприклад, під час натиску клавіш на клавіатурі розробленою програмою відстеження сигналів переривань фіксується декілька значень затримки переривань для усіх можливих станів натиснутої клавіші — натискання, утримання, відпускання.
3. Деякі переривання викликають труднощі у дослідженні. Окрім неможливості відстеження переривань взаємодії з дискетами та модемами у разі відсутності відповідних апаратних компонентів у системі, що досліджується, переривання, сигнал про появу яких відправляється користувачем через його взаємодію з системою, мають великі варіації у часі їх затримки. Так, загальний час затримки переривань, викликані натисканням клавіш на клавіатурі, залежить від часу, витраченого користувачем на відпускання клавіші.

Наразі переривання в ОСРВ QNX також плануються вивчатись на предмет відмінності між результатами вимірів за використання реальної апаратури та віртуальних машин, причини виникнення від‘ємних значень у результатах експериментів та наявності впливу інших програмних одиниць та станів (таких, як паралельні процеси та інші переривання) на часову характеристику затримок переривань різних типів.

Тези до XV Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (16 –17 грудня 2021 року, м. Дніпро), секція «Інтелектуальні інформаційні та телекомунікаційні технології промислових і транспортних систем».

Щодо питання про дослідження впливу затримки переривань на величину найгіршого часу виконання програм

Волкодавець А. О., Нечай В. Я., Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна, Україна

Для задач жорсткого реального часу важливою умовою розв'язку є визначення найгіршого (з точки зору величини) часу виконання програми (Worst Case Execution Time, WCET). На величину цього часу може впливати велика кількість факторів. Найбільш розповсюдженим способом оцінки WCET є статистичні методи, однак ці методи частіше за все дають завищену оцінку. Гібридні методи дозволяють прискорити процес оцінки WCET, але вони не додають точності.

Більш точну оцінку WCET для конкретного класу задач можна віднайти, знаючи складові WCET та їх «внесок» у значення його величини. Справа в тому, що, окрім часу виконання програмних команд, під час роботи програм можуть бути присутні затрати часу на планування потоків та процесів, виконання системних викликів, реалізацію засобів синхронізації потоків, а також затримку переривань. Оскільки операційна система QNX є передбачуваною системою реального часу, вона має засоби визначення даних затрат. Завбачивши, що величини складових WCET мають випадковий характер, для оцінки необхідно отримати їх статистичні параметри.

У даній роботі досліджуються часові характеристики затримки обробки переривань. Вони можуть за часом суттєво перевищувати виконання безпосередньо обробників переривань, однак розглянуті літературні джерела, присвячені проведенню аналогічних досліджень, не зазначають таких особливостей обробки переривань та методологію їх дослідження, тому розв'язок такої задачі подається як досить актуальний.

Величини затримки переривань вимірювались для переривань системного таймера, клавіатури, комп'ютерної миші та контролера жорсткого диску. Були написані функції-обробники переривань за їх відповідними лініями. Мета цих обробників полягала у тому, щоб зафіксувати час отримання сигналу переривань та час виконання першої команди обробника — різниця отриманих часових значень і визначає затримку обробки переривань. Було досліджено вплив середовища виконання замірів на результуючі значення замірів затримки у контексті наступних факторів:

- використання засобів віртуалізації. Розкид значень затримки обробки переривань у випадку використання віртуальних машин є на порядок вищим, ніж у випадку «чистої» QNX — отримане значення стандартного відхилення (величини розкиду значень вибірки відносно математичного очікування) для «чистої» QNX у випадку переривань миші є меншим за стандартне відхилення для віртуальної машини майже у 4 рази;
- завантаженість системи. Як очікувалось, завантаженість системимає слабкий впливна час затримки — різниця значень математичного очікування для вибірок часових значень затримки за нормальних умов та завантаженості системи у більшості розглянутих випадків не перевищує 1 мс. Це пояснюється тим, що порядок виконання задач в ОС QNX залежить від пріоритетів цих задач і переривання, що за архітектурою QNX мають найвищий пріоритет, витісняють інші задачі з нижчим пріоритетом. Для менш пріоритетних сигналів переривань ці закономірності мають більш виражений характер.

Виконані розрахунки часових характеристик та впливу на їх значення різних факторів програмного середовища дозволяють доповнити розрахунки WCET та уточнити значення цієї метрики для конкретних класів задач, розв'язання яких залежить від своєчасної обробки переривань.