Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра кібербезпеки та комп'ютерних систем і мереж

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Модуль формування сигнатур для системи прогнозування взаємоблокувань
Назва теми

КвРКІ. 170347.17.03.58 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія»*
Назва

Виконав: студент IV курсу, група КІ–17–3 _____ О.В.Обозовий
Підпис, Ініціали, прізвище

Керівник _____ С.В. Мостовий
Підпис, дата, Ініціали, прізвище

Нормоконтролер _____ І.В. Муляр
Підпис, дата, Ініціали, прізвище

До захисту допускаю:
Зав. кафедри кібербезпеки
та комп'ютерних систем і
мереж _____ Ю.П. Кльоц
Підпис, Ініціали, прізвище

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем

Кафедра Кібербезпеки та комп'ютерних систем і мереж

Освітній рівень БАКАЛАВР

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри Ю. П. Кльоц

"17" 02 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Обозовому Олексію Вадимовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Модуль формування сигнатур для системи прогнозування взаємоблоквань

Керівник проекту (роботи) Мостовий Сергій Володимирович

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 17.02.2021 № 44 додаток №9

2. Строк подання студентом проекту (роботи) на кафедру 28.05.2021

3. Вихідні дані до проекту (роботи) модуль формування сигнатур для системи прогнозування взаємоблокувань

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Дослідження предметної області та постановка задачі; обгрунтування базових положень щодо проєктування модулю формування сигнатур для системи прогнозування взаємоблокувань; Опис схем електричних (структурної,функційної, принципової) проєктованої системи; опис алгоритму роботи системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Схеми розробленого алгоритму (Е1)

Логічна схема модулю (Е8)

Робоче вікно модулю (Е8)

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Муляр І.В кандидат технічних наук, доцент	-	
Антиплагіат	Муляр І.В кандидат технічних наук, доцент	-	

7. Дата видачі завдання « 19 » 02 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітки
1.	Підготовка вступного розділу	Березень - 1 декада	
2.	Огляд існуючих методів, засобів	Березень - 2 декада	
3.	Обгрунтування обраних рішень	Березень - 3 декада	
4.	Підготовка опису електричних схем	Квітень - 1 декада	
5.	Виконання розрахункової частини	Квітень - 1 декада	
6.	Підготовка ескізів креслень	Квітень - 2 декада	
7.	Формулювання висновків	Квітень - 3 декада	
8.	Розробка додатків	Травень - 1 декада	
9.	Погодження розділів з консультантом з нормоконтролю	Травень - 1 декада	
10.	Оформлення графічного матеріалу	Травень - 2 декада	
11.	Оформлення пояснювальної записки	Травень - 2 декада	
12.	Попередній захист кваліфікаційної роботи	Травень - 3 декада	
13.	Доопрацювання кваліфікаційної роботи	Травень - 3 декада	
14.	Подання роботи для перевірки на плагіат	Травень - 3 декада	
15.	Захист кваліфікаційної роботи	Червень - 1 декада	

Студент Підпис О.В Обозовий Ініціали, прізвище

Керівник проекту (роботи) Підпис С.В. Мостовий Ініціали, прізвище

Тема к
системи прог
Автор р
Керівни
Поясню
Графіч

Метою
прогнозуван

Під ча
формування
взаємоблок
модулю
взаємоблок

Підп

АНОТАЦІЯ

Тема кваліфікаційної роботи: «*Модуль формування сигнатур для системи прогнозування взаємоблокувань*».

Автор роботи: *Обозовий Олексій Вадимович*.

Керівник роботи: *Мостовий Сергій Володимирович*.

Пояснювальна записка: *57 с., 25 рис., 6 табл., 41 джерел*.

Графічна частина: 6 презентаційних слайдів.

Метою даної роботи є розробка модуля формування сигнатур для системи прогнозування станів взаємоблокування

Під час виконання кваліфікаційної роботи роботи, було створено модуль формування сигнатур який призначений для систем прогнозування взаємоблокувань. Новизна даної роботи заключається у створені компактного модулю для подальшого вбудовування в системи прогнозування взаємоблокувань.

_____ _____

Підпис студента Дата

Формат	Позначення	Найменування	Кіл-ть листів	№ екз	Примітка
		Текстовідокументи			
	КвРКІ.170348.17.03.25ПЗ	Пояснювальна записка	57		
		Графічні матеріали			
	КвРКІ. 170348.17.03.25Е8	Логічна схема модулю	1		
	КвРКІ. 170348.17.03.25Е8	Робочевікно модулю	1		
	КвРКІ. 170348.17.03.25Е8	Код в компіляторі Visual Studio	1		

КвРКІ. 170348.17.03.26 ВП

Арк	№ докум	Підпис	Дата
Розроб.	Обозний О.В.		
Перевір.	Косовець С.В.		
Н. контр.	Мулар І.В.		
Затв.	Ключко Ю.П.		

Відомість проекту
Пояснювальна записка

Літера	Аркуш	Аркушів
У	1	1

ХНУ, КІ-17-3

ЗМІСТ

Змн.	Арк.	№ докум.	Підпис	Дата	КвРКІ. 170348.17.03.25ПЗ		
Розроб.		Обозовий О.В.			Модуль формування сигнатур для системипредбаченнявзаємоблокування Пояснювальна записка	Літ.	Арк. 2 / Акрушів 62
Перевір.		Мостовий С.В.					
Н. Контр.		Муляр І.В				ХНУ гр. КІ-17-3	
Затверд.		Кльонін О.П					

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних

ОС - операційна система

ПЗ - програмне забезпечення

BIOS – basicinput / outputsystem , тип вшитого апаратного пристрою в

ПК – персональнийкомп`ютер

МП – мікропроцессор

PID – processidentifier – ідентифікаторпроцесу

PPID – parentprocwssidentifier – ідентифікаторбатьківськогопроцесу

PCB - process control block –управляючийблокпроцесу

ЦП – центральнийпроцесор

ВСТУП

Під час роботи на персональних комп'ютерах інодівиникаютьвипадки з блокуваннямпроцесів. Причиною такиєвипадківявляютьсяпомилки у коді тих чиінших додатків, абоневідповідністьостанніхїхспецифікації. Витікаючою проблемою цього є поломка певнихпрограмабопроблеми з апаратнимискладовими. Для прогнозування та створення сигнатур взаємоблокуваннявикористовуютьбагатоматодів, але не всі з них є практичними у планівикористаннявживу. Головною проблемою є неможливістьїхреалізації в швидкорозвивающихся, сучаснихопераційних системах. Але, тіметодиякіможливореалізуватибудутьзанадто великими відносноопераційноїсистеми, тому в минуломурозробники ОС не встраювалиніякихдодатків для вирішенняцієїпроблеми так, як в тічаси вона буланезначною.

Але з плином часу розвитокопераційних систем зробив великий крок і тепервід ПК потребують великих обсягівобчислювальнихпотужностейц. І самеця потреба почала викликати проблему з взаємоблокування. На сьогоднішній день існуєбагатопрограмякідомомагаютьвідслідковуватиблокуваннячинавітьдопомага ютьїхвирішуватичипопереджати. Такожіснуєдоситьбагатосторонніхпристроїв для цієїзадачі. Але, зсилаючись на те щодосить велика їхкількість, являєтьсянедоцільною у використанні так, як частина з них розроблена для занадтоунікальнихвипадківабонавіть з проблемами в реалізаціїбагатокористувачів не маютьзмогиїхвикористовувати.

Метою моєїроботи є розробка модуля формування сигнатур для системипрогнозуваннястаніввзаємоблокування. Для цьогопотрібно:

1. Провести аналізпроцесів, ти причини їхблокування.
2. Провести аналіз причин взаємоблокувань
3. Проаналізуватиметодивирішеннявзаємоблокувань.
4. Розробити модуль для формування сигнатур у виглядіпрограмногододатку.

1АНАЛІЗ ПРЕДМЕТНОЇ ЧАСТИНИ

1.1 Апаратнезабеспечення

Апаратне забезпечення складається з:

Системнийблок(Рис.1)(Рис.2):

1. Корпус.
2. Процесор.
3. Материнська плата.
4. Внутрішнєсховище(пам’ять).
5. Зовнішнєсховище(пам’ять).
6. Блок живлення.
7. Відеокарта.
8. Звукова карта.
9. Порти введення/виведення.

Рисунок 1.1 - Системний блок

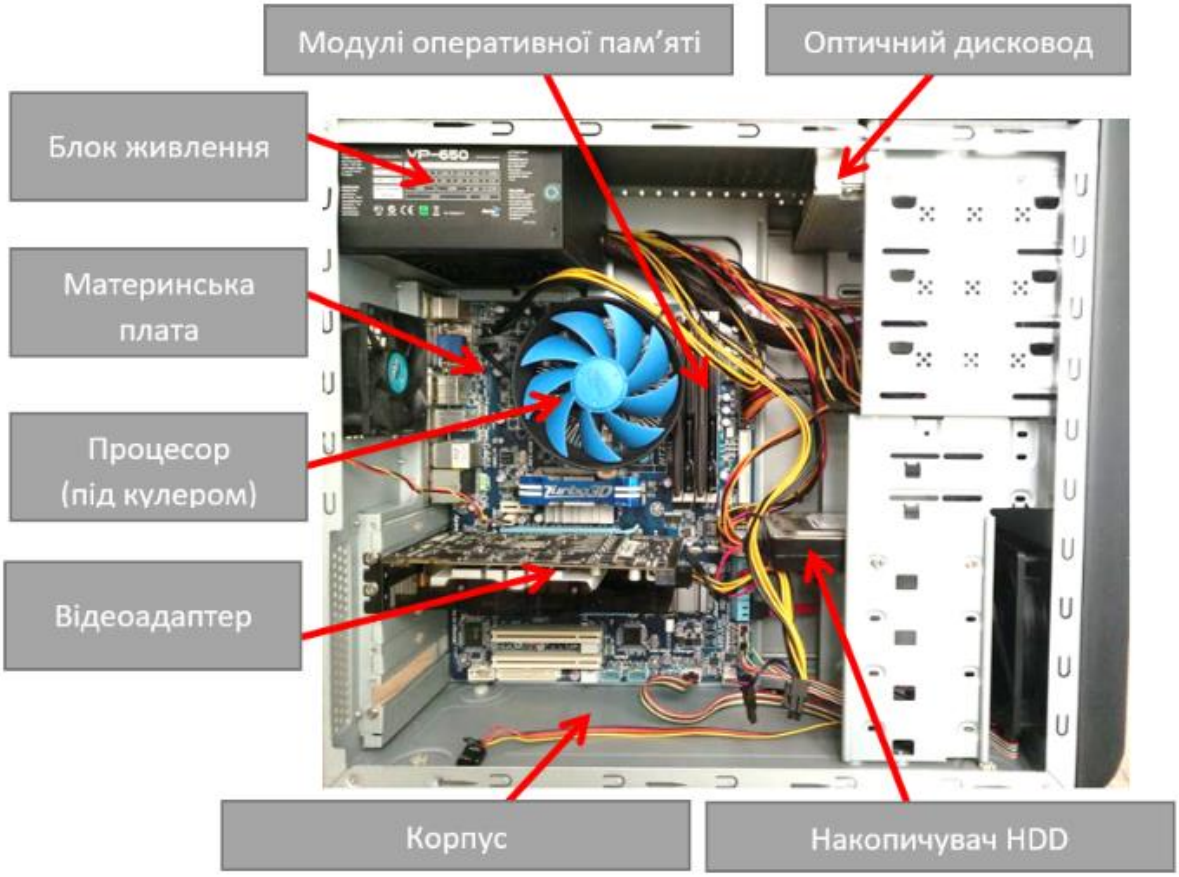


Рисунок 1.2 - Складові системного блоку

1. Обладнання для введення інформації.
2. Обладнання для виведення інформації.
3. Комунікаційне обладнання.

Системний блок (оболонка) Системний блок настільного ПК - це прямокутна рамка, на якій розміщені всі основні компоненти комп'ютера: материнська плата; адаптер, блок живлення, привід гнучких дисків, один (іноді більше) привід, динаміки, компакт-диск диск або інші накопичувачі, елементи керування.

Блок живлення (Рис.3)

Ця машина перетворює потужність змінного струму зі стандартної потужності (220 В, 50 Гц) у низьковольтну потужність постійного струму.Вона має кілька виходів для різних напруг (12 і 5 В), щоб забезпечити живлення відповідної комп'ютерної техніки.

Більшість джерел живлення мають вентилятор для видалення надлишкового тепла з системного блоку, яке виділяється під час роботи електронного пристрою.

Система (материнська плата) Це назва масштабної друкованої плати у стандартному форматі, яка несе основні компоненти комп'ютерної системи: процесор; оперативна пам'ять; кеш-пам'ять; набір логічних чіпів, що підтримують роботу чіпсет материнської плати; центр дороги або шини; контролер шини і кілька знімних слотів для підключення (слотів) для підключення інших плат (контролерів тощо).

Рисунок 1.3 – Блок живлення

Деякі слоти в початковій конфігурації ПК все ще доступні. Модуль оперативної пам'яті встановлений у рознімному з'єднанні інших конфігурацій. Кількість і тип знімних з'єднань є однією з важливих характеристик материнської плати, оскільки вбудованих слотів може бути недостатньо під час добудови або оновлення комп'ютера. Крім того, материнська плата також має мікроперемички або перемикачі для налаштування материнської плати.

На системній платі також є роз'єм, до якого за допомогою спеціального кабелю (кабелю) підключені інші пристрої. Ще одним важливим елементом, встановленим на материнській платі, є мікросхема BIOS(Рис.4). - це енергонезалежний запам'ятовуючий пристрій (ПЗУ), який записує програми, що реалізують функції вводу-виводу, програми, що перевіряють роботу комп'ютера при включенні живлення та інші спеціальні програми. BIOS використовує інформацію про апаратну конфігурацію комп'ютера. Це енергонезалежна пам’ять і постійно живиться від акумулятора на материнській платі. Він також живить кварцовий годинниковий механізм - годинник реального часу, який може постійно відлічувати час і поточну дату.

Рисунок 1.4 – Система Bios

Рисунок 1.5 – Мікропроцессор

Мікропроцесор (МП)(Рис.5), по суті, є мікрокомп'ютером. Основними параметрами МП є набір команд, бітрейт і тактова частота. Набір команд або система постійно вдосконалюються, і з'являються нові команди, які замінили початкову серію команд (мікропрограму). Порівняно з прошивкою, нові команди вимагають меншої кількості циклів. Сучасні депутати можуть виконувати щонайбільше тисячі команд (інструкцій). Функції МР: Біт Тактова частота (Гц); Аудит Розмір кешу (МБ). Бітрейт показує, скільки двійкових бітів (інформації) обробляється (або передається) за тактовий цикл, і скільки двійкових бітів може бути використано в МР для адресування оперативної пам'яті, переданих даних

тощо. Тактова частота вказує, скільки основних операцій виконує МП за секунду, в мегагерцах (1 МГц = 100 000 Гц), і є лише відносним показником продуктивності МП. Перенапруга.

1.2 Процеси та потоки

Під час запуску програми операційна система створює середовище завдань [36], що включає доступ до різних системних ресурсів (пам'яті, пристроїв вводу-виводу, файлів тощо). Це середовище виконання також називається оточеннямта іменується процесом. Коротше кажучи, процес є копією виконуваної програми. Потік - це полегшена версія процесу, послідовність команд, які він обробляє. Один або кілька потоків можуть стати частиною процесу. Потік - це окреме завдання в процесі, тому існує багато відмінностей між процесом і потоком:

Кожен процес має свою пам'ять. Потоки, що працюють усередині нього, ділять пам'ять між собою.

Процеси всередині операційної системи мають власні ідентифікатори. Потік існує в процесі та має ідентифікатор у дії програмного забезпечення.

Кожен потік має власний стек і набір регістрів; завдяки правильній реалізації потоки мають певні переваги перед процесами.

Вони повинні:

1. Менше часу для створення нового потоку, оскільки створюваний потік використовувати адресний простір з поточною діяльністю.
2. Менше часу для його завершення.
3. Менше часу для перемикання між двома потоками в межах одного процесу.
4. Менше комунікаційних витрат, так як потоки поділяють всі ресурси. Використання більше одного потоку є найбільш потужним засобом збільшення швидкості відповіді системи на дії користувача, а також засобом обробки даних, необхідних для одночасного виконання завдання. Такий підхід називається багатопоточність. Він дозволяє зменшити призначене для користувача час

очікування реакції програми на свої дії. Розпаралелювання дозволяє виконувати одночасно велику кількість роботи, що є ефективним засобом при великих обчисленнях.

Багатопоковістьсофту дозволяє створити базу для багатозадачності - виконання декількох завдань одночасно (при багатопроцесорній системі), або «майже одночасно» (при однопроцесорній системі). Наявність декількох потоків надає змогу:

1. Покращитипроцесорганізаціїповедінок програм. Найчастішереакціяпрограмногозабезпеченняорганізовується у декілька незалежних однорівневі (паралельні) алгоритмів, в такому випадкуїхвиносятьта відокремлюють в потік. При цьому їм можна задавати різний пріоритет виконання.

2. Обходити затратні по часу часу операції. Якщо програма однопотокова, в такому випадку вона зупиняєвсіпроцеси при очікуванні довгихопераций, таких як запис уфайлчи програваваннявідео/фото. Втакомувипадкупроцесорзнаходиться в режимі простою, допокиопераціяне закінчиться. У випадкубагатопотоковогододатку, він буде продовжувати свою роботуувідокремих потоках,в той час коли певний потік очікує на завершенняоперації ща сповільнює систему.

3. Відтворити мультипроцесорну обробку. Якщо система, де працює софт, є багатопроцесорною, то ми можемовикористатибагатопоточність для запуску нашоїроботи. Найголовнішещорізні потоки будутьвідтворюватись одночасно на різних процесорах.

Отже, провильнареалізаціябагатопоточності будеполіпшувати використання софту.На одному процесорі багатопотоковістьреалізується за частковоїактивізації різних потоків. Таке перемикання відбувається дужешвидко, для того щобкористувачівідчував роботу потоківмайжеодночасним. У мультипроцесорних і багатоядерних системах потоки будуть і повиннівиконуватисьодночасно. В такому випадкукожний процесор ті ядро, будутьобробляти обробляє відокремлений потік.

Багатопоковісистемитакож добре підтримуютьбагатопроцессів(задач) – цемultitaskingвластивістьопераційноїсистемипідтриматвиконання в

системідекількохпроцесів. Реальна багатозадачність на відмінувід псевдо-паралельномувиконанню, можливалише в розподілених ОС.

Багатозадачністьподіляють на 2 типи:

1. Багатозадачністьпроцесів.
2. Багатозадачністьпотоків.

Найпростішібагатозадачнісередовищанадають та підтримуютьчіткий “розподілресурсів”, коли кожному завданнюсередовищенадаєпевнакількістьпам'яті, а в свою чергкзавданняактивуються через строго визначеніпроміжки часу.

Коли завданняпочинаються в пам'ятіабозалишаютьпам'ять, більшдосконалібагатозадачнісистемидинамічнорозподіляютьресурси на основівсвоїхпріоритетів та системних правил.

Цебагатозадачнесередовищемаєтакі характеристики:

1. Система допомагає в комунікаціїміжпроцесами.
2. Кожному процесунадаєтьсясвійпріорітет, за допомогоюякогоотримує час та пам'ять.
3. Система можеопрацьовуватизапити в реальному часі.
4. Система обов'язково рано чипізноопрацюєвсізадачі.
5. Система реалізуєчергизавдань.
6. Система захищаєодневід одного завдання, при непарнійроботі.
7. Система моніторитьпроблеми в завданнях.
8. Система передбасаєстанивзаємоблокування.
9. Система автоматично перемикаєтьсяміжзадачами(Рис.6).

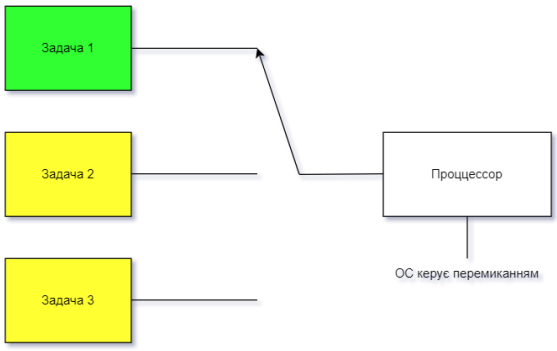


Рисунок 1.6 – Перемиканняміж задачами

1.3 Процесивзаємоблокувань

Процесивзаємоблокувань [10] – цепевнакількістьзаблокованихпроцесів ,кожен з якихбереже ресурс і чекає на отриманняіншого ресурсу, що в той же момент часу зберігаєтьсяіншим ресурсом.

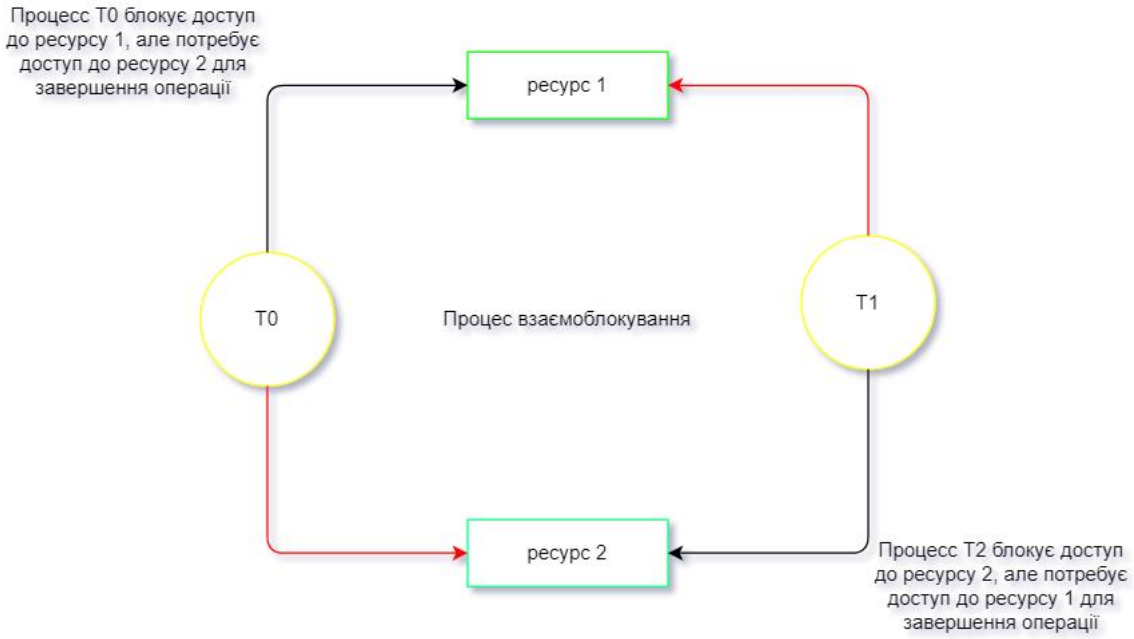


Рисунок 1.7 – Процесивзаємоблокувань

На малюнку вище процес Т0 має ресурс1, для завершення його виконання потрібен ресурс2. Аналогічно, процес Т1 має ресурс2, і йому також потрібно отримати ресурс1, щоб закінчити його виконання.

Таким чином, Т0 і Т1 потрапляють у глухий кут, оскільки кожному з них потрібні ресурси інших для завершення їх виконання, але жоден з них не готовий відмовлятися від своїх ресурсів.

Загалом, процес повинен вимагати ресурс перед його використанням, і він повинен звільнити ресурс після його використання. Одним з головних правил є те, що будь-який процес може вимагати стільки ресурсів, скільки йому потрібно для виконання поставленого завдання. А також, існує умова, що кількість запитуваних ресурсів не може перевищувати загальної кількості ресурсів, доступних у системі.

В основному в звичайному режимі експлуатації використання ресурсів процесом відбувається в такій послідовності:

1. Запит: По-перше, процес запитує ресурс. У випадку, якщо запит не може бути наданий негайно (наприклад: ресурс використовується будь-яким іншим процесом), тоді запитуючий процес повинен почекати, поки він зможе отримати ресурс.

2. Використання: Процес може працювати на ресурсі (наприклад: якщо ресурсом є принтер, то в такому випадку процес може друкувати на принтері).

3. Випуск: Процес звільняє ресурс.

Давайте подивимось на різницю між голодуванням і ситуацією взаємоблокування в таблиці 1.3:

Таблиця 1.1 – Різниця станів Взаємоблокування та голодування.

Голодування	Взаємоблокування
Коли всіпроцеси з низькимпріоритетомблокуються, в той час як процеси з високимпріоритетомвиконуються, цяситуаціяназивається - голодуванням.	Взаємоблокування - цеситуація, яка виникає, коли один ізпроцесівзаблокований.

Закінченнятаблиці 1.1

Голодування - цеочікування(коротке/довге), але це не безкінечнийпроцес.	Взаємоблокування - ценескінченнийпроцес.
Не обов'язково, щобкожноголодуваннябуловзаємоблокуванням.	У кожному взаємоблокуванні є голодування.
Голодуваннязумовленененеконтрольованимпріоритетом та управлінням ресурсами.	Під час ситуаціївзаємоблокуванняпопередження та круговеочікування не відбуваютьсяодночасно.

Виникнення ситуаціївзаємоблокування може виявити планувальник ресурсів. Тепер розглянемо умови, які необхідні для створення ситуації взаємоблокування.

Ситуація взаємоблокування може виникнути лише за умови, що всі наступні чотири умови виконуються одночасно:

1. Взаємне виключення. Згідно з цією умовою, принаймні один ресурс повинен бути недоступним (нерозподіляються ресурси, які можуть використовуватися одним процесом за раз).
2. Утримання та очікування. Відповідно до цієї умови, процес утримує щонайменше один ресурс і чекає додаткових ресурсів.
3. Відсутність попередження. Ресурси не можна брати з процесу, оскільки ресурси можуть бути звільнені лише добровільно тим, хто їх тримає.
4. Кругове очікування. У цьому стані набір процесів чекає один одного в круговій формі.

Вищезазначені чотири умови не є повністю незалежними, оскільки умова кругового очікування передбачає умову утримання та очікування.

Потрібнонаголосити на тому, що всі чотири умови повинні виконуватись у взаємоблокуванні.

Уникнути умов тупикової ситуації можна за допомогою ряду методів. Розглянемо деякі методи.Методи, які використовуються для вирішення проблеми тупикових ситуацій, такі:

1. Ігнорування ситуації взаємоблокування.

Згідно з цим методом, передбачається, що мертвої точки ніколи не відбудеться.

Цей підхід використовується багатьма операційними системами, де вони припускають, що стан взаємоблокування ніколи не відбудеться, що означає, що операційні системи просто ігнорують стан взаємоблокування.

Цейпідхідможе бути корисним для тих систем, яківикористовуютьсялишедля перегляду та для звичайнихзавдань. Таким чином, метод ігноруваннявзаємоблокуванняможе бути корисним у багатьохвипадках, але він не ідеальний для того, щобусунути стан взаємоблокуванняізопераційноїсистеми.

2. Запобігання замкнутості.

Як вже задувалось у наведеному вище розділі, усі чотири умови: взаємне виключення, утримання та очікування, відсутність попередження та циклічне очікування, якщо система утримує, викликають стан взаємоблокування.

Основною метою методу запобігання взаємоблокування є порушення будь-якої однієї умови серед чотирьох; тому що, якщо будь-яка з однієї умови буде порушена, проблема взаємоблокування ніколи не виникне. Отжеідеяцього методу проста, але труднощіможутьвиникнутипід час фізичноїреалізаціїцього методу в системі.

3. Уникнення взаємоблокування.

Цей метод використовуєтьсяопераційною системою для того, щобперевірити, чиперебуває система в безпечномучинебезпечномустані. Цей метод перевіряєкожен крок, виконанийопераційною системою. Будь-якийпроцеспродовжуєсвоєвиконання, поки система не перейде в безпечний стан.

Як тількисистема переходить у небезпечний стан, операційна система повинна зробити крок назад.

В основному, за допомогоюцього методу операційна система слідкує за кожнимрозподілом і переконайтеся, щорозподіл не спричиняєжодноїситуаціївзаємоблокування в системі.

4. Виявлення та відновлення блокування.

За допомогоюцього методу взаємоблокуваннявиявляєтьсяспочатку за допомогоюдеякихалгоритмівграфікурозподілуресурсів. Цейграфік в основному використовується для представленнярозподілурізнихресурсів для різнихпроцесів. Післявиявленняситуаціївзаємоблокування, можнавикористати ряд методів для відновлення з цього стану.

Одним ізспособів є випередження, за допомогоюякого ресурс, якийутримується одним процесом, надаєтьсяіншомупроцесу.

Другийспосіб - цевідкат, оскількиопераційна система веде запис про стан процесу, і вона може легко повернутипроцес до попереднього стану, завдякичомуситуаціювзаємоблокуванняможливо легко усунути.

Третійспосібподоланняситуаціївзаємоблокування - цевбивство одного абодекількохпроцесів.

1.4 Програми для моніторингупроцесів ОС

В даний час існуєвеличезнакількістьрізнихдодатків, створених з метою моніторингуресурсів ОС. Найвідоміші з них:

1. SystemExlorer(Рис.8).

SystemExplorer – безкоштовний додаток для ОС Windows,який являє собою диспетчер задач, але з набагато більшим функціоналом. Ця программа допомагаєкористувачувідстежувативсіактивніпроцеси в системі, а також детально відображаєзавантаженняпам’яті і файлу підкачки. Також в програмувбудована база даних для перевіркифайлів на шкідливий код. System

explorerдозволяєкористувачузакінчитипроцесабо групу процесівякщо вони нагружають систему.

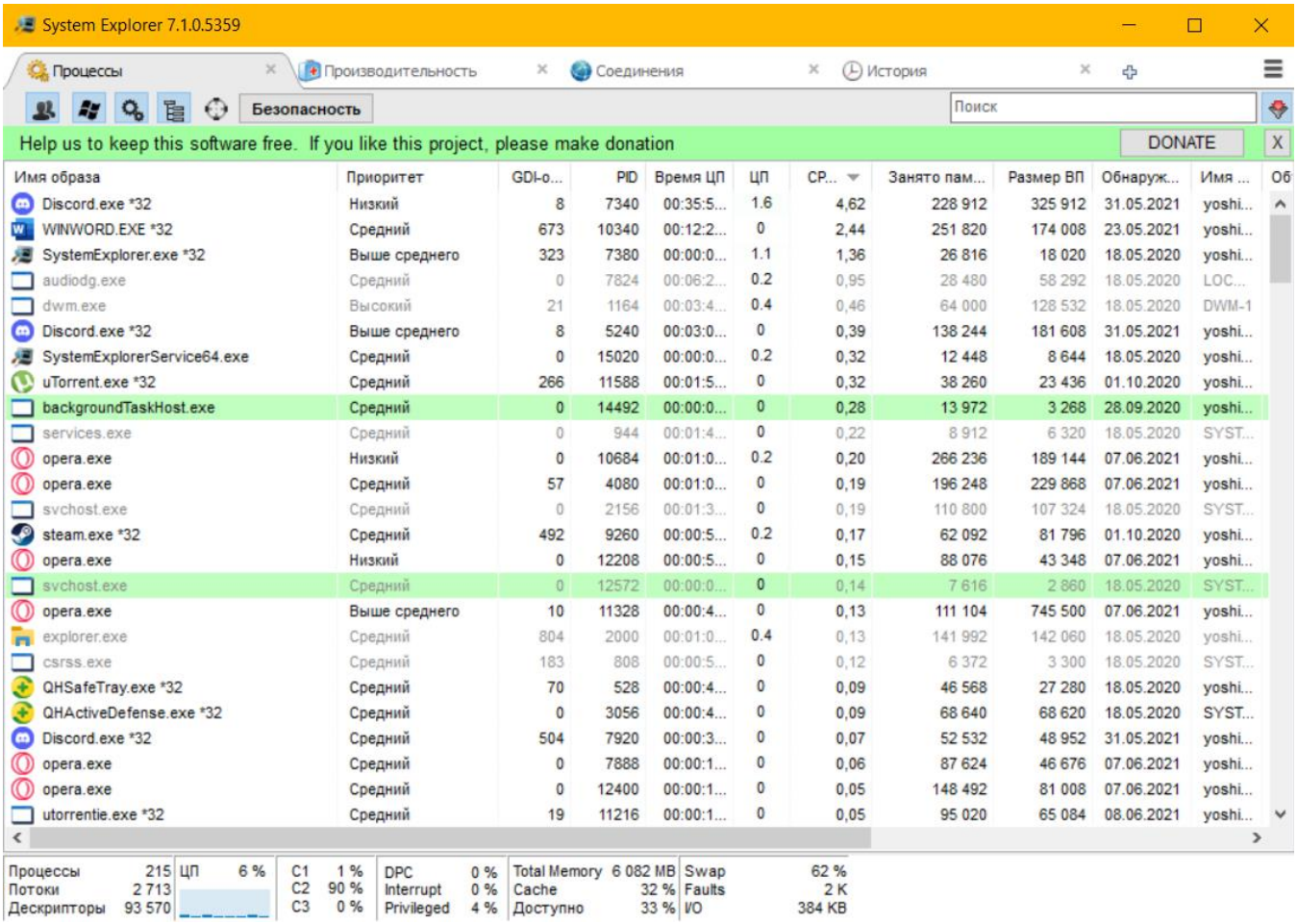


Рисунок 1.8 – System explorer

2. Securitytaskmanager(Рис.9).

Securitytaskmanager - це програма, яка аналізує запущені процеси для виявлення потенційно шкідливих процесів. Програмапризначаєпевнийрівеньпотенційноїнебезпеки (від 0 до 100) для кожного запущеногопроцесу. Той самий рейтинг базується на відгукахгромади. Ви також можете перевірити будь-якийпроцесзі списку послугVirusTotal і дізнатисякількістьпозитивнихвідповідейвідрізнихантивірусних програм. За допомогоюцієїпрограмиви можете визначити, чиіснуєпроцес, якийвідображає рекламу в системі, проводить майнінг криптовалют у фоновому режимі, передаєконфіденційнуінформацію про користувачів,

допомагаєвіддаленокеруватикомп’ютерами та робить інші «неприємніречі». Менеджер завданьбезпеки особливо чутливий до непідписанихпроцесів та процесів, яківикористовуютьнадмірнісистемніресурси.

Щеоднієюцікавоюособливістюпрограми є те, що вона можеперевірятиіншіаспектисистемноїбезпеки, наприклад, чипрацюєантивіруснепрограмнезабезпечення, та збереженіфайлисookie, якіможнавикористовувати для відстеженнякористувачів. Інтерфейс диспетчера завданьбезпекискладається з вікназі списком процесів. Вибравши будь-який з них, вивідкриєте панель, щоміститьдетальнуінформацію про сертифікати, пов’язані з програмою та розробникамикомпанії. Двічіклацніть, щобвідкритивікно, де можназалишитиособистийкоментар.

Додатоктакожможеоцінитибезпекувашогоантивірусногопрограмногозабезпечення за допомогоювласного рейтингу та рекомендуватибільшнадійніпрограмицього типу, якщо рейтинг нижчий.

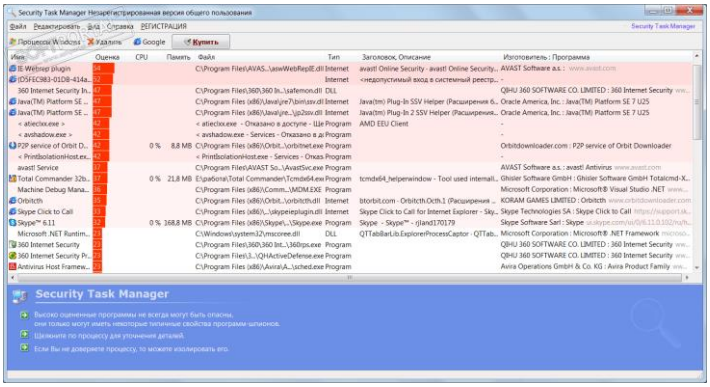


Рисунок 1.9 - Security task manager

3. Moo0 SystemMonitor(Рис.10).

MooOSystemMonitor - надає змогу відслідковувати системні ресурси ПК. На сьогоднішній день додатокнадаєзмогу, отримувати 43 типи інформації, а аключаючи ЦП та Пам'ять. А такожнадаєповнуінформацію по використаннюжорсткого диску.

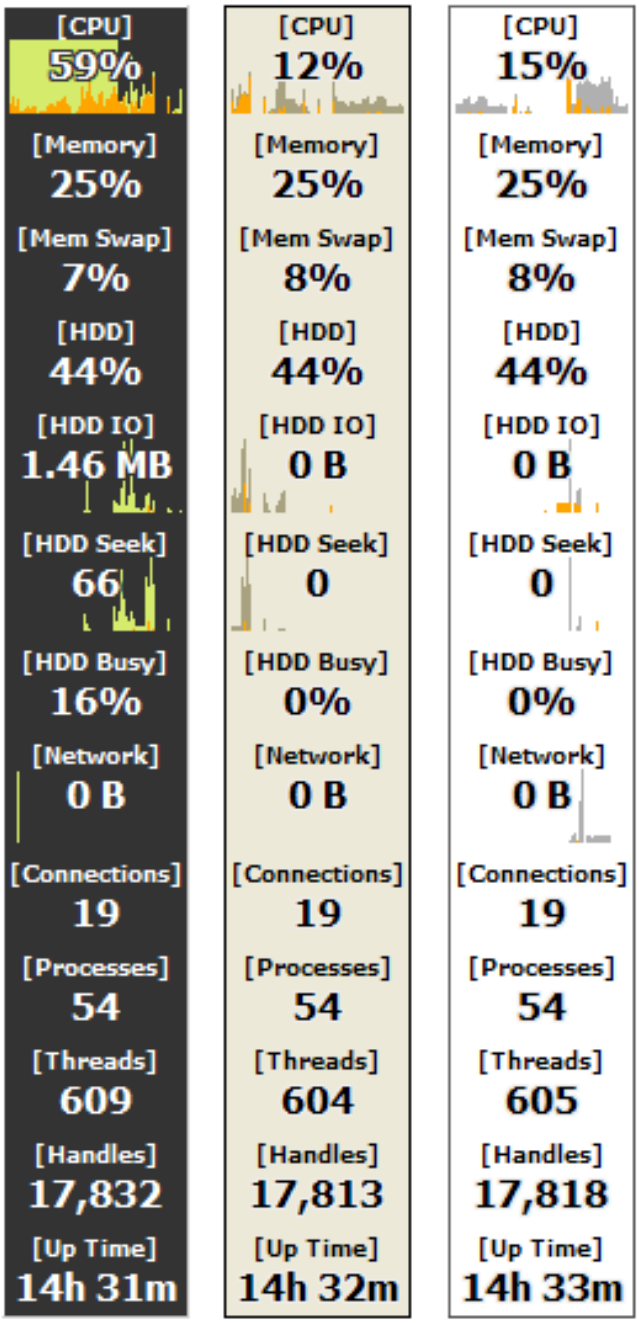


Рисунок 1.10 - MooO System Monitor

4. FarbarRecoveryScanTool(Рис.11).

FarbarRecoveryScanningTool - це портативний інструмент, який може збирати детальну інформацію про систему та різні програми/послуги. Програма готує детальні звіти про всі процеси та файли, щоб допомогти виявити наявність шкідливих програм на комп'ютері користувача. Користувачам доведеться вирішити більшість виявлених проблем самостійно, цей інструмент може лише скасувати зміни, внесені до певних системних файлів, та виконати інші

виправлення. Досвідчені користувачі можуть створювати власні відредаговані файли та "подавати" їх на інструмент сканування відновнику Farbar.

Зазвичай такі програми використовуються для надання віддаленої допомоги у видаленні системних інфекцій.

FarbarRecoveryScanTool - це портативна програма, яку можна запускати зі змінних носіїв. Він також може працювати в "безпечному" режимі. Після завершення сканування програма збереже заповнений звіт у тій самій папці, де зберігається виконуваний файл. Farbar повністю безкоштовний.

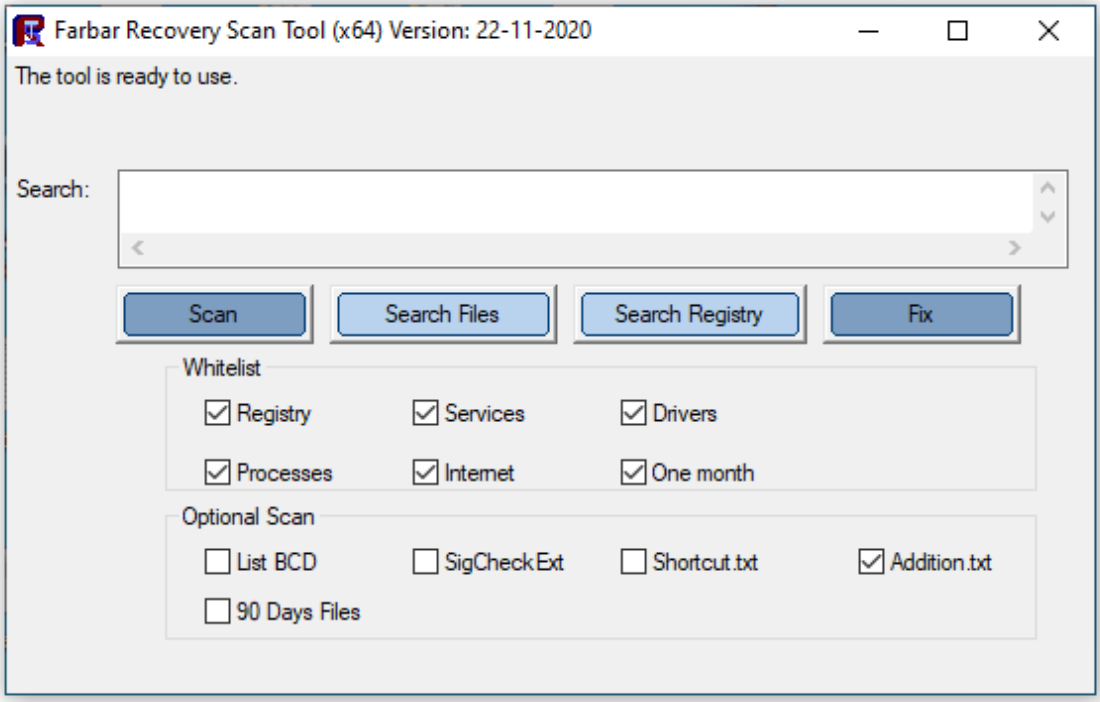


Рисунок 1.11 - Farbar Recovery Scan Tool

Це лише мала частина розроблених додатків. Кожне з них підходить тільки під певні типи процесорів і метринских плат. Часто буває, що при установці на комп'ютер утиліти не можуть отримати доступ до ресурсів ОС в результаті закритих налаштувань адміністратора.

До того ж більшість додатків є вузькоспеціалізованими. Існують утиліти для тестування клавіатури, оптимізації відеокарт, вимірювання продуктивності жорстких дисків і їх дефрагментації, очищення операційної пам'яті і ін.

Для отримання інформації про системні ресурси ОС Windows в наш час використовують вбудовану утиліту: “Диспетчер завдань” (Рис.12)

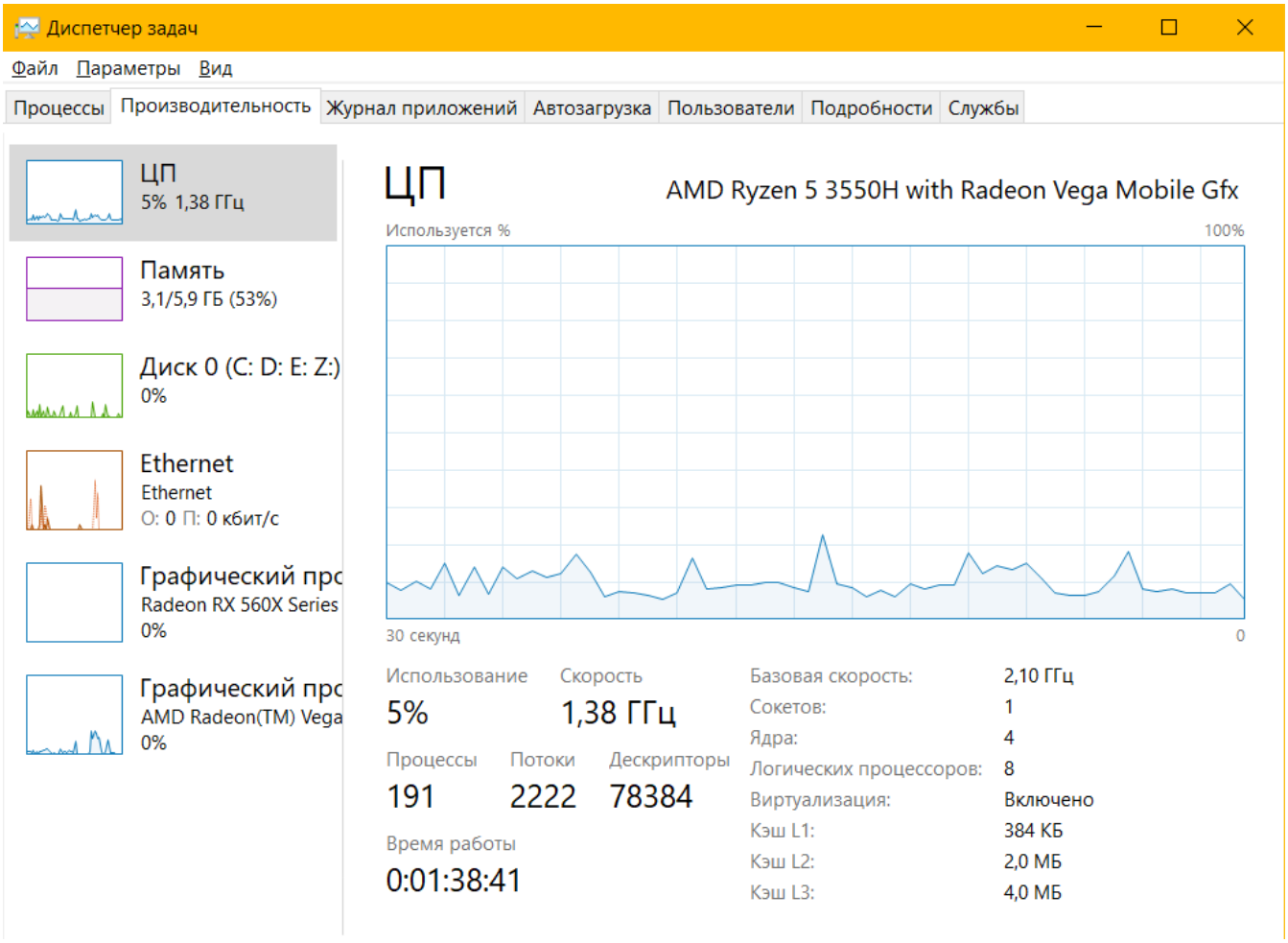


Рисунок 1.12 - Диспетчерзавдань

1.5 Постановка задачі

По попереднійінформаціїпотрібновідслідковвувати та отримуватиінформацію про системніресурси ОС. Основні вимоги- це отримання параметрів про певніапаратнічастини. Більшість з цихутилітможутьпрацюватинекоретно через те що вони можуть не підходити до певнихкомпонентів ОС.

Завданнями програмного забезпечення є:

1. відстеження робочих процесів.
2. відслідковування стану всіх ядер ЦП, а саме: отримання данних про завантаження та частоту.
3. відслідковування стану оперативної пам'яті.

Отже, нам потрібнодинамічноотримуватиотримувативузьконаправленіданні.

1.6 Висновокпершогорозділу

Отже, в цьомурозділібулорозібрноскладові аппаратного забеспечення ПК, а такожпоннятяпроцесів та взаємоблокувань. Поямненоосновніпроблеми та перевагипевнихпрограм для відслідковуванняпроцесів та навівприкладиостанніх.

А токожзробленовисновки по темі причини взаємблокування та визначеннямożливихваріантівїхвирішення.

2ПРОЦЕСИ ТА РЕСУРСИ ОС

2.1 Сигнатурипроцесів

Прогнозування стану процесу в ПК ґрунтується на контролі та визначенні моменту, коли процеснайбільшнаближенийдо станувзаємноблокування, за допомого поточного реэструванняпоказникывпроцесу

Означимопараметри та всіймовірні характеристики процесу, якимивінвиражається для побудовийогосигнатури.

Коли користувачзапускаєпроцес, тимщозапускаєпевнупрограму в операційнійсистемі. Запущенніпроцеси не враховуються. Програма сама по собі- цепрограмний код та деякідані,щознаходяться в файліпрограмичи в даннихоперативнійпам'яті ОС якібудутьвиконуватись. Операційна система реалізуєокремийнабірданних(контекст процесу), для кожного нового процесу. Інформацію в цьомунаборііменують атрибутами процесу - задлянаступногопокращеннярозрізнення одного процесувідіншого. Останніможузалишатисянезмінним/змінюватисяпротягомжиттєвого циклу процесу

Найважливішіатрибутицьогопроцесувключають [7]:

1. Ідентифікаторпроцесу (ІдентифікаторпроцесуPID)(Рис.1). Ідентифікаторомпроцесуобов'язковоповинне бути ціле, додатнє число. А такожусіпроцеси у ОС повиннніматиунікальнийідентифікатори. Операційна система використовуєідентифікаторипроцесу для управлінняостанніми.

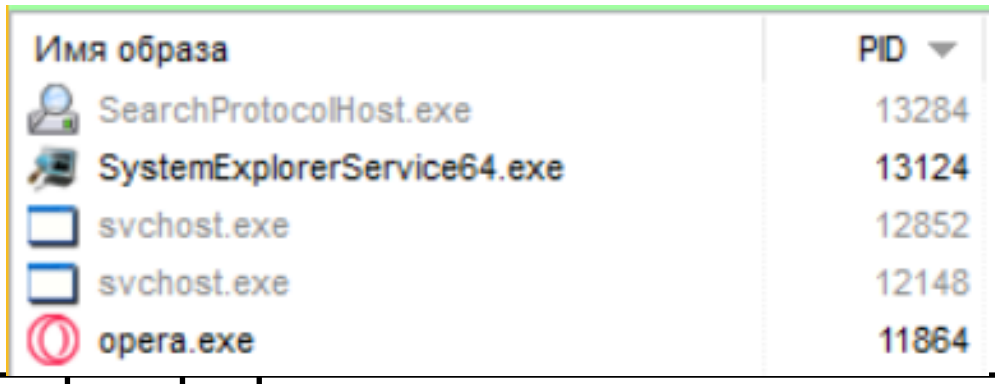


Рисунок 2.1 – ІдентифікаторPID

2. Ідентифікаторбатьківськогопроцесу (PPID). Цей атрибут під час початку роботиотримуєтьсяпроцесом та можевикористовуватись для йоговиклику, подання сигналу на ньогочиотриманняінфориації про нього.

3. Реальні та ефективніідентифікаторикористувачів та груп (UID, 47 GID, EUID та EGID)(Рис.2). Реальні та ефективніідентифікаторикористувачів та груп, дужесхожі з ідентифікаторомкористувачаякийрозпочинавпроцес, та групи, якійвінналежить.

Реальніідентифікаторикористувачівуспадковуєтьсядочірнімпроцесом, і заблоковані до їхзміни. Доступ до комп'ютернихресурсіввизначається за допомогоюефективнихідентифікаторів.

Рисунок 2.2 – Ідентифікаторигруп

4. Відкрити файл. На додаток до стандартнихфайлівведення, виводу та помилоктакийпроцесможевідкрити й іншіфайли. Дочірнійпроцесуспадковуєвсіфайлищобуливідкритічивикористанібатьківським фалом;

5. Пріоритет та відноснийпріоритет(Рис.3). Пріоритетпроцесуявляєтьсязмінною величиною щопрораховуєтьсяпід час

виборувиконуваногопроцесу. Пріорітетзалежатимевідфакорів часу очікування, поточного стану та відносногопріорітетупроцесу.

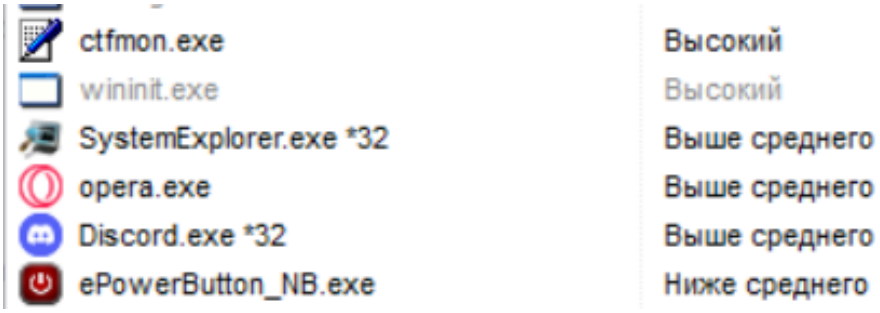


Рисунок 2.3 – Приорітет процессу

6. Поточний каталог. Усіпроцеси в системіповинніматисвоїпоточні і робочі каталоги.

7. Часу обробки.

8. Змінніседовища.

9. Розмірпрограми - обсягпам'яті, зайнятийпроцесом. Усіпроцеси в системіозначаютьсяданними про розмір/резиденськийрозмір.

Ціпараметри є загальними для більшості ОС (існуютьвинятки). Однакцей список можнадоповнитидеякимиіснуючими параметрами, щоможуть бути головними для тоїчиіншоїопераційноїсистеми. Для вирішенняпроблемипрогнозування стану процесу ми пропонуємовизначимоостанній за допомогоюпевного набору характеристик (сигнатуру).

Сигнатурою процесів називатимемонабірінформації про його характеристики, щобудутьвиділятипевнийпроцес в системі.

Для виділення характеристик процесупроцесу з подальшимформуваннямсигнатури, бутьвідноситисьтакідані(приклад процесів на Рис.4):

1. Ідентифікаторпроцесу.

2. Батьківський/користувацькийідентифікатор.

3. Квота процесу (пам'ять, приоріетність).

4. Дескрипторищовикористовуєпроцес.
5. Кількістьвикористаноїпроцесомвіртуальноїпам'яті.
6. Час якийпотребуєпроцес для виконання.
7. Іншіпараметри.

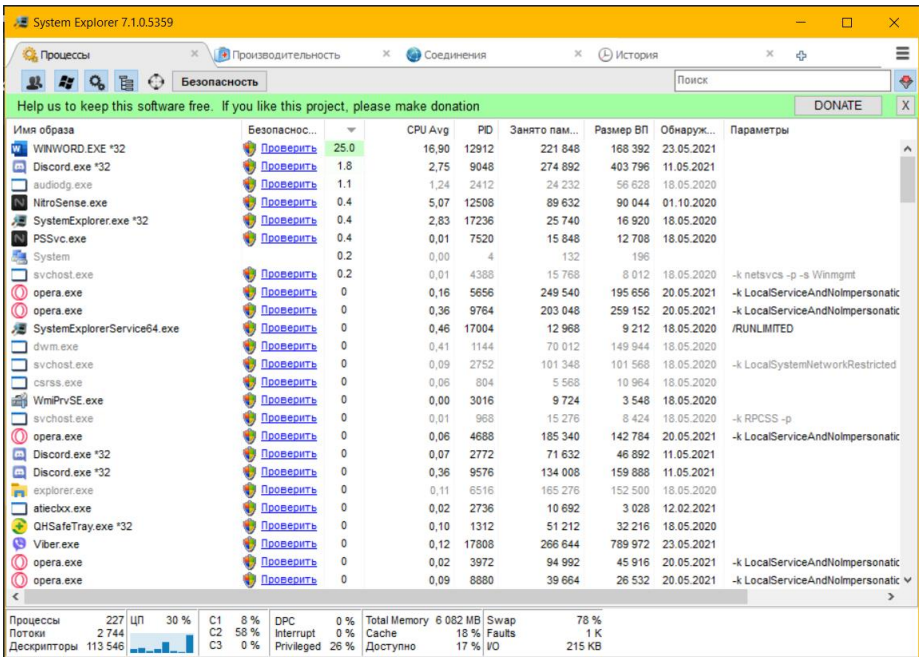


Рисунок 2.4 - Приклад запущенихпроцесів

А також, занесемоцідані в таблицю(Таблиця 2.1)

Таблиця 2.1 – Ідентифікаторипроцесу.

Параметр	Кількістьбітів	Використання системою
Ідентифікаторпроцесу	14 біт.	+
Батьківський/користувацькийідентифікатор	14/4 біт.	+
Квота процесу (пам'ять, приоріетність)	30/4 біт.	+
Дескрипторищовикористовуєпроцес	8 біт.	+

Кількістьвикористаноїпроцесомвіртуальноїпам'яті	30 біт.	+
Час якийпотребуєпроцес для виконання	30 біт.	+
Іншіпараметри	-	+/-

1.2 Життєвий цикл процесів

Программа [36] що в певний момент часу виконуєтьсяпристроєм, називаютьпроцес. Розумінняпроцесумаєпевнукількістьскладових(Рис.5).

Перше, цепослідовневідпрацювання команд якіактивує алгоритм програми.

Друге - повторно використовніресурси, щонадаютьсяпроцесу. Це оперативна та віртуальнапам'ять, канали вводу/виводу та відкритіпрограмоюфайли.

Третє - наповненнярегістрівпроцесора та значеннявнутрішніхзміннихпрограмияківціломупоказуютьвнутрішній стан процесу.

В кінцікінців, цезначеннязовнішніхвідноснопроцесузмінних, яківиділяютьпроцес та надані для ньогоресурси і фіксують стан процесузісторони ОС.

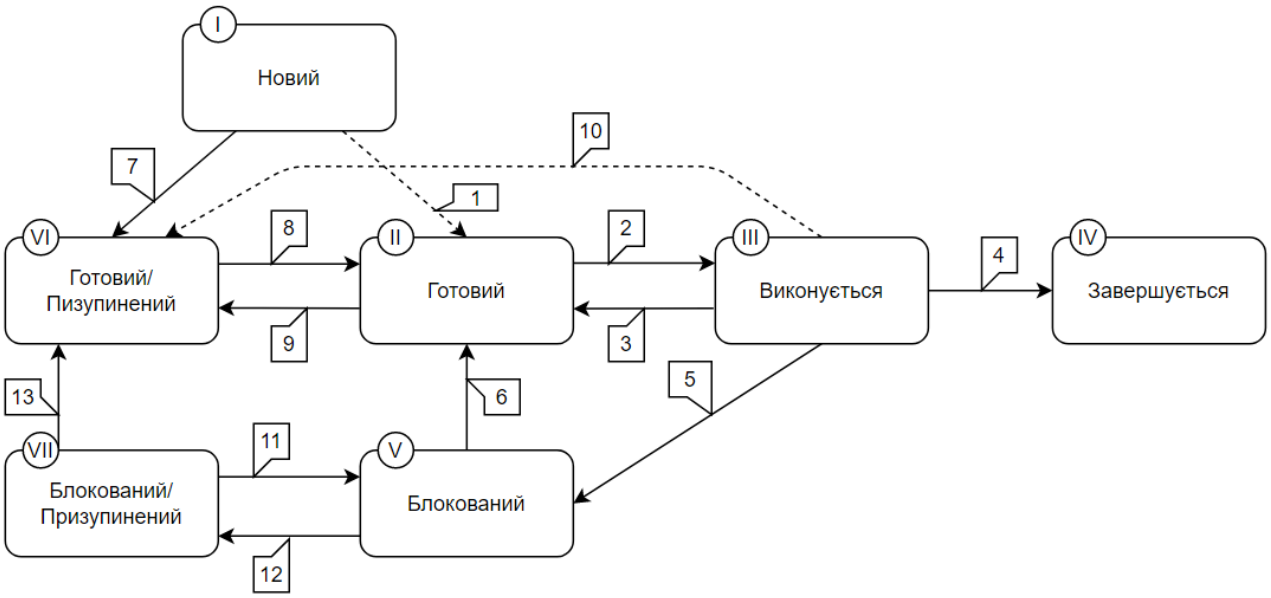


Рисунок 2.5 - Життєвий цикл процесів

Цізміннінайчастішеявляютьсячастинамиоперативних структур даних ОС. Життєвий цикл кожного процесунайкраже буде показати за допомогоюдіаграми(Рис.10) на якійзображеністанипроцесу та переходи поміж ними. На діаграмізображено увесь життєвий цикл процесівзісімома станами. Розглянемоособливості кожного стану. Розглянемопоетапнокожнийпроцес:

1. Перший процес (I) створюється ОС (може бути створений по командіїншогопроцесу) для реалізаціїдеякоїпрограми. В цей момент система створює та зберігає в пам'ятіідентифікаториданих, щовиділяютьпроцес, зберігаютьдані про його стан та містятьінформаціюпотрібну для управління та використанняпроцесом. Ця дана інформаціяназиваютьтерміномуправляючий блок процесу (processcontrolblock – PCB). Також, операційна система маєрозмістити в пам'яті код програми і дані, а найголовніше, повинна надатидостатньопам'яті для стековякіреалізовуються для запусків процедур.

2. У випадкуякщо, в даний момент ресурсів в системідостатньо, абонавітьбільше для виконання то процес переводиться в стан “готовий” до виконання (II).

У випадкуякщосистемнихресурсівнедостатньо для активаціїпроцесу, то вінзаморожуєтьсядо моментупокиопераційна система не виділитьресурси. В цей

момент операційна система реалізуємеханізмсвопінгупідкачки(Рис.18) для перенесення образу процесу на диск, а при наданніресурсів – обратного повернення ресурсу в оперативнупам'ять.

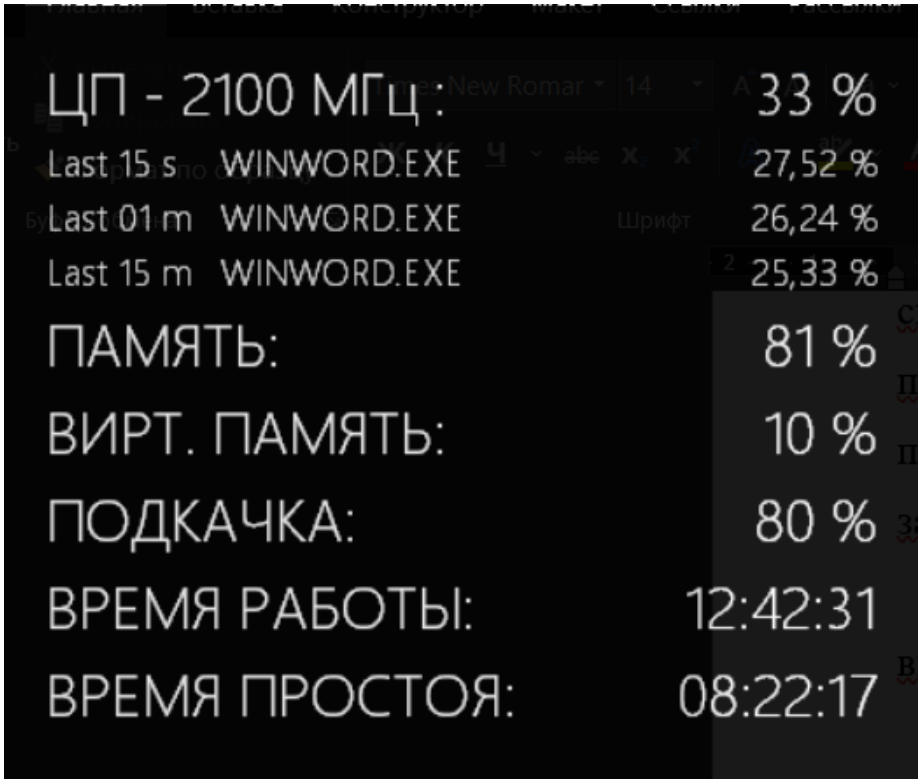


Рисунок 2.6 - Приклад підкачки

3. Звідсибачимо переходи 8-9. Тобто при наявностіресурсівопераційна система переводить процесиміж режимами готовий/призупинений – готовий.

4. Всіпроцеси, якіготовідо запуску, знаходяться в упорядкованійчерзі і час від часу передаютьсяпланувальником у стані "працює" (III) (компонент ОС, який по черзізабезпечуєпроцеспроцесором). Цеймеханізмтакожназиваютьперемиканнямпроцесів. У цьомувипадкупроцесотримаєцентральнийпроцесор, який буде виконуватипрограму до кінцявідведеногоперіоду часу, абопоки не буде подано запит на процесобслуговування ОС, результат результату не може бути в поточному періоді часу Всередині. Наприклад, компіляторперетворюєопераціювведенняданихмовоюпрограмуваннявисокогорівн

я у виклик до відповідноїслужби ОС (системнийвиклик), і заздалегідьневідомо, коли данінасправдідоступні в програмі в наступнихпрограмах. Почнітьдрукувати.

5. Тому менеджер звільняєпроцесор і переводить процес у «заблокований» стан (V). У цьомустаніпроцестриватиме до надходження асинхронного повідомлення (час прибуттяякогоневідомий), щовказує на те, щопотрібніданідоступні в ОС і операціявведенняможе бути завершена. Повідомленняобробляєтьсяопераційною системою, щоб перевести процес у готовий стан. Якщо в заблокованомустанібагатопроцесів, а є процеси в стані "готовий / призупинений", операційна система можеперерозподілитиресурси для передачіпевнихпроцесівіз стану "готовий / призупинений" у "готовий" стан. Видалітьдеякізаблокованіпроцеси. Останнібудутьзнаходитись у "заблокованому / призупиненому" стані (VII) до настанняочікуваноїподії.

6. Оскількизаблоковані та заблоковані / призупиненіпроцесичекаютьпоявиасинхроннихподій, їхчергивпорядковані, тоді як чергиготових та готових / призупиненихвпорядковані. Тип сортуваннярізнихчергможе бути різним, і операційна система визначаєтьсястратегієюпланування (загальний план діяльності, для досягненняскладної мети потрібентривалий час, в даномувипадку - максимальна продуктивністьсистеми) та плануваннямпроцесу.

7. Результатом закінченняпрограми є перехідпроцесуіз стану виконанняу стан «повного» (IV). Це не обов'язковоозначає, щопроцесвидаляєтьсяізсистеми. Зазвичай образ такого процесуміститьінформацію, необхіднуіншимпроцесам. У цьомувипадкучастинаструктуридрукованої плати залишатиметьсяпротягомпевногоперіодупісляпроцесу. У серії UNIX стан цьогопроцесуназивається "зомбі".

На графіку є кільказагальнихкоментарів (Рис17). На схеміпоказанівсіможливістанипроцесу та можливі переходи між ними.

У той же час кожен стан можематибезлічпроцесів, окрім «запущеного», у цьомустані в комп’ютернійсистемінемаєбільшепроцесів, ніжпроцесори. У цьомувипадкучастковіперетворення (1, 2, 6-13) ініціюються ядром ОС і не

маютьнічогоспільного з текстом запущеноїпрограми. Перетворення (3) зазвичайздійснюєтьсяпланувальником в кінціобраного часового квантового процесу. Це так звана превентивна багатозадачність, реалізована в більшостісучаснихопераційних систем загальногопризначення.

У спільнійбагатозадачніймоделіпроцес сам ініціюєперетворення (3) і виконуєсистемнийвиклик для звільненняпроцесора.

Перехід (4) відповідаєвиконанню системного виклику для завершенняпроцесу. Цеможе бути явно позначене як викликпроцедури в програмі, абовономоже бути неявно позначене, коли досягнутозакінченняпрограми. Ви також можете примусовопринитипроцесіншимпроцесом з відповіднимидозволами.

1.3 Механізмизахистуресурсів

Неважкозробитивисновки - глобальнізмінні ,щовідокремлені (тобтоявляютьсяспільними для використаннядекількомапроцесами), такожповинні бути захищені. Єдинийспосіб для цього – керування кодом, якиймаєповнийресурсний доступ до них.

Але, для подальшоїкоректноїроботиважливощобпроцеси могли взаємодіяти з ресурсами які вони поділяютьпоміж собою.

Видивзаємодіїпроцесіввраховуючиїхобізнаність про існуванняіншихможнакласифікувати за допомогоютаблиці 2.2.

Таблиця 2.2 – Рівеньобізнаностіпроцесів.

Рівеньобізнаності	Тип взаємодії	Впливпро

Процесамнаданаінофрмація один про одного.	Процесибудутьвзаємодіятиміж собою.	Один з процесі але можли процесівзалежа
Процесамчасковонаданаінформація один про одного.	Процесибудутьвзаємодіятиміж собою, але ділитиресурси.	Один з процесі але можли процесівз
Процесам не наданаінформація один про одного.	Процесибудутьконкуруватиміж собою за отримання ресурсу.	Один з процесі але не зале

Продовженнятаблиці 2.2

У випадку, якщо, процесу не наданаінформація про інші, він буде використовуватиусіресурси. Такийпроцес буде боротись за повний доступ до ресурсів.

Єдинийхтойогозупинить - операційна система. Підчасборотьби за ресурси ми стикаємосьізтрьома проблемами:

1. Необхідністьвзаємноговиключення – дія коли один ресурс вимикає доступ до ресурсівіншимпроцесам, при умовіщовінотримав доступ перший. Цей ресурс називаєтьсякритичним ресурсом, частинапрограми, яка використовує ресурс критичнимрозділом. Найголовнішещоб в любий момент часу, в критичному розділізнаходиласьлише одна програма. Реалізаціявзаємовиключеньстворюєдвііншіпроблеми.

2. Deadlockабовзаємнеблокування. Через паралельневиконнання задач інодівиникаютьситуації, при якихпроцеси весь час залишаютьсязаблокованими, очікуючиресурсиіншогопроцесу, якийаналогічнознаходиться в заблокованомустані. Такіпроцесиназивають – процесаммивзаєболокуванняабоdeadlockprocces. На сьогоднішній день розглядаєтьсявеличезнакількістьрішень та засобівборотьби з взаємоблокуванням. Але на одному рівні з цимтакожрозглядаєтьсяціна таких проектів тому, що для розробкисистемивільноївідвзаємоблокування буде витрачено велика кількістькоштів.

Якщопроцесамчастковонаданаінофрмація один про одного, цеможе привести до співпраціпроцесів. У ходіякоїпроцесибудутьділитисистемніресурсипоміж собою.

У випадкуякщопроцесамнаданаповнаінформаці один про одного (включаючипосиланнячиназву), вони можутьконтактуватипоміж собою. Такіпроцесинайчастішестворюються для спільноїроботитизалежать один від одного. Найчастішеїхспілкуванняслугує для синхронізуванняданнихчикоординіціїїхдій. Підпоннятямзв'язкурозуміється – відправленняпевнихтипівфайлівміжпроцесами.

При такійроботіпроцесівможливообійтись без взаємовиключення, але найголовнішіпробемивзаємоблокувань та голодування не зникають тому, що два процестиможутьзаблокуватисьпід час одночасногоочікавння листа один від одного.

Підсумовуючивище сказану інформацію, зрозумілощо без взаємовиключенняобійтись не можливо. Але хоч-яка можливістьпідтримкивзаємовинятків повинна підходитинаступнимвимогам:

1. Примусовевиконаннявзаємовиключень.
2. Перериванняроботи поза критичнимрозділомпроцесом, не мусить негативно відображатись на іншихпроцесах.
3. У випадкувідсутності будь-якогопроцесу в критичному розділі, потрібнонегайновідправититудипроцес.
4. Заборонено робитиприпущення про кількістьабовідноснушвидкістьвиконанняпроцесу.
5. Процесмуситьзалишатись у критичному розділілишеобмеженийінтервал часу.

2.4 Висновок другого розділу

У другому розділібулоповноціннорозглянутопроцеси та ресурси ОС що вони використовують. В подальшомуцяінформаціявикористовувалась для вираженняжиттєвого циклу процесів та ресурсівякі вони використовуютьпід час циклу. Післяцього, на підставіідентифікаторівбуло сформовано сигнатуру процесів, а такожзображеномеханізмзахистуресурсів ОС. Вся розглянутаінформаціядаєзмогуотриматиповноцінну картину про причиннуформуваннявзаємоблокуваньпроцесів, що в подальшомудопоможе в розробці модулю формування сигнатур для системипрогнозуваннястаніввзаємоблокувань.

Отже в другому розділібулазібрана та описана вся інформаційна база, що буде повноцінновикористовуватись в наступномурозділі при реалізації коду для модулюформування сигнатур.

3 РЕАЛІЗАЦІЯ МОДУЛЯ ФОРМУВАННЯ СИГНАТУР

3.1 Середовище .NET

.NET Framework (Рис.1) [25] - цетехнологія, яка підтримуєствореннята виконання веб-служб та програм Windows. Під час розробки .NET Framework булирозглянутінаступніцілі:

1. Забезпечитипослідовнеоб’єктно-орієнтованесередовищепрограмування для локального зберігання та виконанняцільовихкодів, локального виконанняабовіддаленоговиконаннярозподіленихкодів в Інтернеті.
2. Мінімізуватиможливістьконфліктівпід час розгортанняпрограмногозабезпечення та контролю версій.
3. Забезпечитибезпечневиконання коду, включаючи код, згенерованийневідомимиабо не повністюдовіренимистороннімипостачальниками, та усунутипроблемиізсередовищемсценаріївабоінтерпретацією коду.
4. Надаєуніфікованіпринципирозробки для різнихтипівпрограм, таких як програми Windows та веб-програми.
5. Забезпечитивзаємодію на основігалузевихстандартів, Цезабезпечуєінтеграцію коду .NET Framework з будь-якиміншим кодом.

Рисунок 3.1 – Microsoft .NET

.NET Framework [1] складаєтьсяіззагальномовноїсередовищавиконання (CLR) та бібліотекикласів .NET Framework. Основою .NET Framework є середовище CLR. Середовищевиконанняможнарозглядати як агента, якийуправляє кодом під час виконання та забезпечуєосновніпослуги, такі як управлінняпам'яттю, управління потоками та віддаленувзаємодію. У цьомувипадкунавколишнєсередовищевстановлюєсуворіумови типу та інші типи перевіркиточності коду для забезпеченнябезпеки та надійності. Насправдіголовнимзавданнямсередовищавиконання є управління кодом.

Код, якийотримує доступ до середовищавиконання, називаєтьсякерованим кодом, а код, який не обертаєсередовищевиконання, називаєтьсянекерованим кодом. Бібліотекакласів - цевсебічнаоб'єктно-орієнтованаколекціябагаторазовихтипів, щовикористовуються для розробкидодатків, відзвичайнихдодатків командного рядка до додатківграфічногоінтерфейсукористувача (GUI), до додатків, щовикористовуютьнайновішітехнології ASP .NET (Рис.2) [26], таких як веб-форми та веб-служби XML.

Рисунок 3.2 – ASP.NETCore

.NET Framework можерозміщуватинекерованікомпоненти, якізавантажують CLR у власнийпроцес і запускаютькерованевиконання коду, створюючитим самим програмнесередовище, щодозволяє як кероване, так і некерованевиконання. .NET Framework не тількизабезпечуєдекількаосновнихсередовищвиконання, але такожпідтримуєрозробкусторонніхсередовищвиконання.

Наприклад, ASP.NET розміщуєсередовищевиконання та забезпечуємасштабованесередовище для керованого на сервері коду. ASP.NET працюєбезпосередньоізсередовищемвиконання, щобзабезпечитизастосуванняпрограм ASP.NET та веб-служб XML, про якійдетьсядалі в ційстатті. Internet Explorer можнавикористовувати як приклад некерованоїпрограми (як розширення типу MIME) під час керованогосередовищавиконання.

Розміщеннясередовищавиконання в Internet Explorer дозволяєвбудовуватикерованіелементикерування Windows Formsабоелементикерування в документи HTML. Церозміщеннядозволяєвиконуватикерованиймобільний код і насолоджуватисяйогосуттєвимиперевагами, включаючивиконання в умовахнеповноїдовіри та ізольованогозберіганняфайлів.

Особливостісередовища CLR

CLR керуєпам'яттю, виконаннямпотоків, виконанням коду, перевіркамибезпеки коду, компіляцією та іншимисистемними службами. Ціінструменти є внутрішніміінструментами для керованого коду, щопрацює в середовищі CLR.

З міркуваньбезпекикерованим компонентам присвоюєтьсярізнийступіньдовіризалежновідбагатьохфакторів, включаючиїхпоходження (наприклад, Інтернет, корпоративна мережа аболокальнийкомп'ютер).

Цеозначає, щокерований компонент можеабо не можевиконуватиоперації доступу до файлів, операцій доступу до реєстручиіншихважливихфункцій, навітьякщовінвикористовується в тому самому активному додатку.

Середовищевиконаннятакожзабезпечуєнадійність коду, реалізуючисуворуінфраструктуру набору та перевірки коду, яка називається Common Type System (CTS). Загальна система типівзабезпечуєсамоописусьогокерованого коду.

Різнікомпіляторимов Microsoft та сторонніхрозробниківстворюютькерований код, якийвідповідаєзагальним типам систем. Цеозначає, щокерований код можеприйматиіншікеровані типи та екземпляри, забезпечуючи при цьомудостовірність типу та суворевведення тексту.

Крім того, Managed Executive усуваєбагатозагальних проблем ізпрограмнимзабезпеченням. Наприклад, виконавчімеханізми автоматично контролюютьрозміщення та зв'язуванняоб'єктів, звільняючиїх, коли вони вже не використовуються.

Автоматичнеуправлінняпам'яттюусуваєдвінайпоширенішіпомилкипрограми: витокипам'яті та недійсніпосилання на пам'ять.

Середовищевиконаннятакожпідвищуєефективністьрозробників. Наприклад, програмістиможутьвикористовуватизнайомімовирозробки для написанняпрограм та повноцінновикористовуватисередовищавиконання, бібліотекикласів та компоненти, написаніншимимовамиіншимирозробниками.

Це доступно для кожного виробникакомпілятора, якийвирішуєпроблемипід час виконання. Мовнийкомпілятор .NET Framework робить .NET Framework доступним для існуючого коду, написаноговідповідноюмовою, значнополегшуючипроцесміграціїіснуючихпрограм.

Хоча Executive розроблений для програмногозабезпеченнямайбутнього, вінтакожпідтримуєпрограмнезабезпеченнясьогоднішнього та вчорашнього часу. Взаємодіякерованого та некерованого коду дозволяєрозробникамвикористовуватинеобхідні COM-компоненти та бібліотеки DLL.

Середовищевиконанняпризначене для підвищенняпродуктивності. Незважаючи на те, що Common Language Executive надаєбагатостандартних служб виконання, віннікоди не пояснюєкерований код. Компіляція на вимогу (JIT) дозволяєвиконувати весь контрольований код машинною мовоюкомп'ютера, на якомувінпрацює.

У той же час менеджер пам'ятівиключаєможливістьфрагментаціїпам'яті та збільшуєобсягпам'яті для подальшогопідвищенняпродуктивності. Нарешті, Executive можнарозмістити на високопродуктивнихсерверних додатках, таких як Microsoft SQL Server та IIS (ІнформаційнаслужбиІнтернету).

Цяінфраструктурадозволяєписатилогікупрограми за допомогоюкерованого коду,
одночасновикористовуючивисокупродуктивністьнайкращихвиробничихсерверів, щопідтримуютьрозгортанняпід час виконання.

3.2 Реалізація модулю формування сигнатур

Поняття "процесу" існувало в операційних системах Windows задовго до появиплатформи .NET. Попросту кажучи, підпроцесомрозуміється програма, щовиконується.

Однак формально процес - цеконцепціярівняопераційноїсистеми, яка використовується для опису набору ресурсів (таких як зовнішнібібліотеки коду і головнийпотік) і необхідноїпам'яті, використовуваноїщовиконуютьсядодатком.

Для кожного завантаження в пам'ять файлу * exe в операційнійсистеміствориться окремий ізольованийпроцес,
якийвикористовується на протязівсього часу йогоіснування.

Завдякитакійізоляціїдодатків,
виконуючогосередовищавиходитьнабагатобільшнадійної і стабільної, оскількивихід з ладу одного процесуніяк не позначається на роботіінших процесів.

Більш того, доступ безпосередньо до даних в одному процесі з іншогопроцесунеможливий, якщотільки не застосовується API-інтерфейсрозподіленихобчислень, такий як Windows Communication Foundation.

Через всіхцихмоментівпроцесможевважатисяфіксованою і безпечної кордоном виконуєтьсядодатки.

Коженпроцес Windows отримуєунікальнийідентифікаторпроцесу (Process ID - PID) і моженезалежнозавантажуватися і розвантажуватисяопераційною системою (в тому числіпрограмно).

Як вже напевно відомо, у вікніWindowsTask Manager (Диспетчер завдань) є вкладка Processes (Процеси), на якійможнапереглядатирізністатичнідані про виконуються на даніймашиніпроцесах, в тому числіїх PID-ідентифікатори та іменаобразів.

Щобвідкритивікно диспетчера задач (Рис.3), натиснітькомбінаціюклавіш<Ctrl + Shift + Esc>:

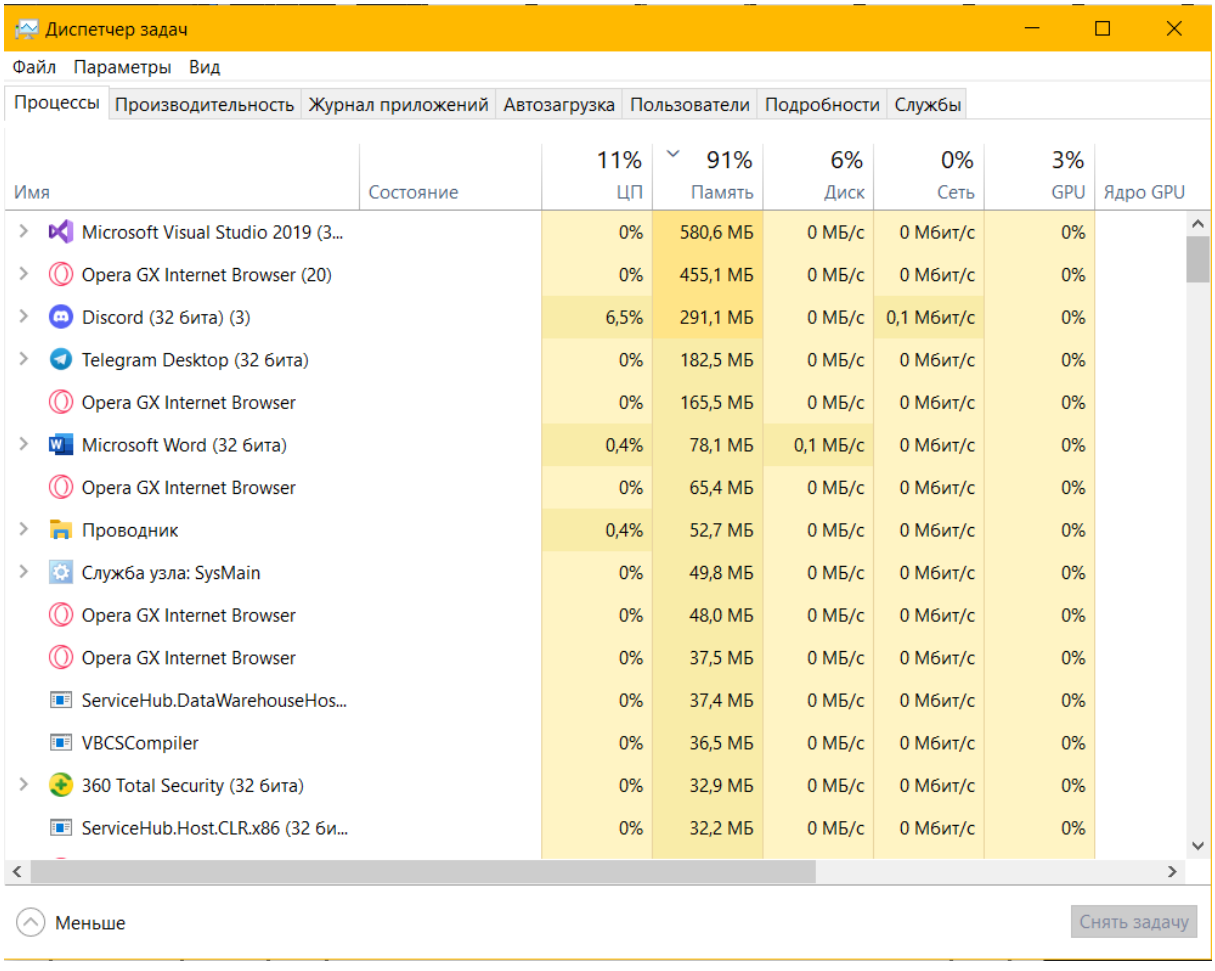


Рисунок 3.3 – Диспетчер задач

Роль потоків

У кожному процесі Windows міститьсяпервинний "потік", який є вхідний точкою для додатка.

Потоком називаєтьсявикористовуванийвсерединіпроцесу шлях виконання. Формально потік, якийстворюється першим у вхіднійточціпроцесу, називаєтьсяголовним потоком (primarythread). У будь-якийвиконуванійпрограмі .NET (консольному додатку, додатку Windows Forms, додатку WPF і т.д.) вхідні точка позначається як метод Main (). При викликуцього методу головнийпотікстворюється автоматично.

Процеси, в якихміститьсяєдинийголовнийпотіквиконання, спочатку є безпечними до потокам (threadsafe), оскільки в коженокремий момент часу доступ до данихдодатка в них можеотримуватитільки один потік.

Однакподібніоднопоточніпроцеси (особливо з графічнимпризначеним для користувачаінтерфейсом) часто уповільнених реагують на діїкористувача, коли їхєдинийпотіквиконуєякусьскладнуоперацію (зразоквиведення на друкдовгого текстового файлу, складнихматематичнихобчисленьабоз'єднання з віддаленим сервером).

Через такого потенційногонедолікуоднопоточнихдодатків, API-інтерфейс Windows (а також платформа .NET) надаєможливість для головного потоку породжуватидодатковівторинні потоки (такожзваніробочими потоками).

Це робиться іззастосуванням набору функцій з API-інтерфейсу Windows, таких як CreateThread (). Коженпотік (первиннийабовторинний) в процесістаєунікальним шляхом виконання і можепаралельноотримувати доступ до всіхрозділяютьсяелементамданихвсерединівідповідногопроцесу.

Як неважкоздогадатися, розробникизазвичайстворюютьдодаткові потоки для поліпшеннязагальногоступеняприйнятливостіпрограми до дійкористувача. Багатопотоковіпроцесизабезпечуютьілюзію того, щовиконаннячисленнихдійвідбуваєтьсяприблизно в один і той же час.

Наприклад, додатковийробочийпотікможепороджуватися в додатку для виконанняякоїсьтрудомісткоюзавдання (подібноївисновку на друк великого текстового файлу). Після початку виконаннязавданнявторинним потоком основнийпотік все одно не втрачаєздатностіреагувати на діїкористувача, щодаєвсьомупроцесуможливістьсупроводжуватисякудибільшвисокоюпродуктивністю.

Однак такого може і не відбуватися: в разівикористаннязанадтовеликоїкількостіпотоків в одному процесійогопродуктивністьможенавітьпогіршуватися через виникнення у ЦП необхідностіперемикатисяміжактивними потоками в процесі (щовіднімаєпевний час).

На деяких машинах многопоточность здебільшогоявляє собою не більшеніж просто забезпечуєтьсяопераційною системою ілюзію.

Машини з одним (не підтримуютьгіперпотокі) процесором буквально не маютьніякоїможливостіоброблятибезлічпотоків в точності в один і той же час. Замістьцього вони виконують по одному потоку за одиницю часу (звану квантом), грунтуючисьчастково на пріоритеті потоку.

Післязакінченнявиділеного кванта часу виконанняіснуючого потоку призупиняється для наданняіншому потоку можливостівиконатисвоєзавдання. Щобпотік не забував, на чомувінпрацював перед тим, як йоговиконаннябулопризупинено, кожному потоку надаєтьсяможливістьзаписуватидані в локальнесховищепотоків (Thread Local Storage - TLS) і виділяєтьсяокремий стек викликів.

Хоча в самих процеси і потоки немаєнічого нового, спосіб, яким з ними можнавзаємодіяти в рамках платформи .NET, досить пристойно змінився (в кращу сторону).

Щобпрокластисобі шлях до розумінняприйомівпобудовибагатопоточнихзбірок, почнемо з того, щоподивимося, яким чином можнавзаємодіяти з процесами за рахунокзастосуваннябібліотекбазовихкласів .NET.

У просторііменSystem.Diagnosticsпоставляєтьсянабіртипів, якідозволяютьпрограмновзаємодіяти з процесами і різнимипов'язаними з діагностикоюзасобами на кшталт системного журналу подій і лічильниківпродуктивності. Нас цікавлятьтількиті типи, якідозволяютьвзаємодіяти з процесами.

Деякінайбільшважливі з них перерахованінижче: Process Надає доступ до локальних і віддаленихпроцесів, а такождозволяєзапускати і зупинятипроцесипрограмним чином ProcessModuleЯвляє модуль (* .dllабо * exe), який повинен завантажуватися в певнийпроцес.

Важливорозуміти, щоцей тип можезастосовуватися для подання будь-якого модуля - COM, .NET аботрадиційногодовічного на базі C ProcessModuleCollectionДозволяєстворювати строго типізовануколекціюоб'єктівProcessModuleProcessStartInfoДозволяєвказуватинабір значень, якіповиннівикористовуватисяпри запускупроцесу за допомогою методу Process.Start () ProcessThreadПредставляєпотіквсерединіпевногопроцесу.

Слідмати на увазі, щоцей тип застосовується для діагностики набору потоків в процесі, але не для відгалуженнявсерединіньогоновихпотоківProcessThreadCollectionДозволяєстворювати строго тіпізованнуюколекціюоб'єктівProcessThreadКласSystem.Diagnostics.Processдозволяєаналізуватипроцеси, яківиконуються на якійсьпевніймашині (локальноїабовіддаленої).

Крім того, в ньому є члени, якідозволяютьпрограмнозапускати і зупинятипроцеси, переглядатипріоритетпроцесу, а такожотримувати список активнихпотоківабомодулів, якібулизавантажені в даний процес.

У таблиці 3.1 перерахованідеякінайбільшважливівластивості і методикласуSystem.Diagnostics.Process:

Таблиця 3.1 – МетодикласуSystem.Diagnostics.Process.

Метод	Опис
ID	Дозволяєотримуватиідентифікатор (PID) відповідногопроцесу
MachineName	Дозволяєотриматиназвукомпютера, на якомувиконуєтьсяпроцес
ProcessName	Дозволяєотриматиназвупроцесущоспівпадає з назвоюдодаткукотриййого запустив
StartTime	Дозволяєотримувати час коли бувзапущенийдодаток
GetCurrentProcess()	Цейстатичний метод повертаєProcess, якийявляєтьсяактивним в певний момент
GetProcesses()	Цестатичний метод повертаємножинувсіхProcess, щовиконуються в даний момент на машині

Продовженнятаблиці 3.1

Крім повного списку всіх виконуються на конкретній машині процесів, статичний метод Process.GetProcessById () дозволяє отримувати інформацію і по конкретному об'єкту Process за рахунок вказівки асоційованого з ним ідентифікатора (PID). У разі запиту неіснуючого PID генерується виключення ArgumentException. Набір потоків представляється у вигляді суворо тіпізованной колекції ProcessThreadCollection, в якій міститься певна кількість окремих об'єктів ProcessThread.

Властивість Threads в System.Diagnostics.Process надає доступ до класу ProcessThreadCollection, деякі найбільш цікаві члени якого перераховані нижче:

1. CurrentPriorityДозволяєотриматиінформацію про поточнийпріоритет потоку.
2. Id - дозволяєотриматиунікальнийідентифікатор потоку.
3. IdealProcessor - дозволяєвказатибажанийпроцесор для виконанняданого потоку.
4. PriorityLevel - дозволяєотриматиабозадатирівеньпріоритету потоку.
5. ProcessorAffinity - дозволяєвказатипроцесори, на якихможевиконуватисявідповіднийпотік.
6. StartAddress - дозволяєдізнатися, за якоюадресою в пам'ятіопераційна система викликалафункцію, яка призведадо запускуданого потоку.
7. StartTime - дозволяєдізнатися, коли операційна система запустила потік
8. ThreadState - дозволяєдізнатисяпоточний стан даного потоку.
9. TotalProcessorTime - дозволяєдізнатися, скількивсього часу даний потіквикористовувавпроцесор.
10. WaitReason - дозволяєдізнатися причину, по якійпотікзнаходиться в станіочікування.

Перш ніжрухатисядалі, необхідночіткоусвідомити, що тип ProcessThread не є сутністю, яка застосовується для створення, припинення і знищенняпотоків в .NET. Вінскоріше є засобом, щодозволяєотримуватидіагностичнуінформацію по активним потокам Windows всерединівиконуєтьсяпроцесу.

Теперподивимося, як реалізуватипрохід по завантаженихмодулів, якіобслуговуються в рамках конкретного процесу. Під час обговоренняпроцесів модуль - цезагальнийтермін, якийвикористовується для описузаданоїзбірки * .dll (або * exe), яка обслуговується в певномупроцесі.

При отриманні доступу до ProcessModuleCollection через властивістьProcess.Modulesможнавитягти список всіхмодулів, якіобслуговуютьсявсерединіпроцесу: .NET, COM і традиційнихзаснованих на C бібліотек. І, нарешті, останніми членами класуSystem.Diagnostics.Process, якізалишилосярозглянути, є методи Start () і Kill ().

Ціметодидозволяють, відповідно, програмнозапускати і завершуватипроцес. Метод Start () можеприймати тип System.Diagnostics.ProcessStartInfo і надаватидодатковіфрагментиінформаціїщодо запуску певногопроцесу.

Прогнозування стану процесу в ПК ґрунтується на контролі та визначенні моменту, коли процеснайбільшнаближенийдо станувзаємноблокування, за допомого поточного реєструванняпоказниківпроцесу

Означимопараметри та всіймовірні характеристики процесу, якимивінвиражається для побудовийогосигнатури.

Коли користувачзапускаєпроцес, тимщозапускаєпевнупрограму в операційнійсистемі. Запущенніпроцеси не враховуються. Програма сама по собі-цепрограмний код та деякідані,щознаходяться в файліпрограмичи в даннихоперативнійпам'яті ОС якібудутьвиконуватись. Операційна система реалізуєокремийнабірданних(контекст процесу), для кожного нового процесу. Інформацію в цьомунаборііменують атрибутами процесу - задлянаступногопокращеннярозрізнення одного процесувідіншого. Останніможутьзалишатисянезмінним/змінюватисяпротягомжиттєвого циклу процесу

Найважливішіатрибутицьогопроцесувключають [7]:

1. Ідентифікаторпроцесу (Ідентифікаторпроцесу PID). Ідентифікаторомпроцесуобов'язковоповинне бути ціле, додатнє число. А

такожусіпроцеси у ОС повиннніматиунікальнийідентифікатори. Операційна система використовуєідентифікаторипроцесу для управлінняостанніми.

2. Ідентифікаторбатьківськогопроцесу (PPID) - цей атрибут під час початку роботиотримуєтьсяпроцесом та можевикористовуватись для йоговиклику, подання сигналу на ньогочиотриманняінфориації про нього, але в системіWindowsнеможливорозділятиPID та PPID так, як система автоматично надаєбатьківський процесс з йогоідентифікатором.

3. Ідентифікаторкористувача - цей атрибут використовується системою для ідентифікаціїкористувачащо запустив процесс.

4. Обсягпам`яті – це атрибут щовикористовується системою для видачіпам`яті для певного процессу та повертається у виглядіцлого числа.

5. Пріорітетністьпроцесу – цу атрибут використовуєтьсясистеиою для систематичного та послідовного запуску абовиконанняпроцесів.

Ціпараметри є загальними для більшості ОС (існуютьвинятки). Однакцей список можнадоповнитидеякимиіснуючими параметрами, щоможуть бути головними для тоїчиіншоїопераційноїсистеми. Для вирішенняпроблемипрогнозування стану процесу ми пропонуємовизначимоостанній за допомогоюпевного набору характеристик (сигнатуру).

Сигнатурою процесів називатимемонабірінформації про його характеристики, щобудутьвиділятипевнийпроцес в системі.

Для виділення характеристик процесупроцесу з подальшимформуваннямсигнатури (Рис.4), бутьвідноситисьтакідані:

1. Ідентифікаторпроцесу (x_1 біт).
2. Батьківський/користувацькийідентифікатор ($x_2 - x_1$ біт).
3. Ідентифікаторкористувача ($x_3 - x_2$ біт).
4. Пам'ятьпроцесу ($x_4 - x_3$ біт).
5. Приорітетнісьпроцесу ($x_5 - x_4$ біт).
6. Дескрипторищовикористовуєпроцес ($x_6 - x_5$ біт).
7. Кількістьвикористаноїпроцесомвіртуальноїпам'яті ($x_7 - x_6$ біт).

8. Час якийпотребуєпроцес для виконання ($x_8 - x_7$ біт).

9. Іншіпараметри ($x_n - x_8$ біт).

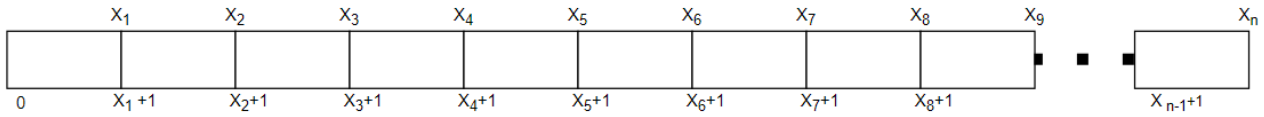


Рисунок 3.4 – Формуваннясигнатури

Розмірсигнатуринапрямузадежитимевідкількостіпараметрів, щобудутьвключені до неё, а такожархітектурніособливостіконкретноїопераційноїсистеми. Для того, щобпоказатипроцеси та відповіднісигнатури, ми виразимоїх у наступномувигляді: (T,S,I), де I - ценазвапроцесу, а S - налаштована сигнатура у цьомупроцесія, T - час коли відбуласьреалізаціясигнатуриабо час останньоїмодифікації для певногопроцесупроцесу.

Побудуємотаблицю 3.2 в якузанесемосаметікритеріїякібудемовикористовувати для будуваннясигнатури.

Таблиця 3.2 – Побудовасигнатури.

№ п/п	Параметр	Команда для виклику	Приклад значеннящоповертає
1	Назвапроцесу	p.ProcessName	opera11748yoshimoto43472 3110656860165True Discord6916yoshimoto8869 4253859845True
2	Ідентифікаторпроцесу	p.Id	
2	Ідентифікаторкористувача	Environment.UserName	
3	Приорітетність	p.BasePriority	
4	Кількістьдескрипторів	p.HandleCount	
5	Обсяг пам`яті	p.VirtualMemorySize64	

Продовженнятаблиці 3.2

6	Час роботои	Environment.TickCount64 / 3600000	
7	Активність	p.Responding	

Для побудовипрограмиспочаткупобудуємологічну схему (Рис.5) для подальшоговикористання.

Послідовністьдій модулю:

1 При запуску додатку першим чином маєрозпочатисьзбірінформації про всіактивніпроцеси в операційнійсистемі, самецюможлиавстьнадає.NET .

2 Наступним кроком коли додатокотримаєінформацію про всіоперативніпроцеси, почнеотримуватидодатковіданні про кожен з них що в подальшомудопоможесформувати сигнатуру Цядія буде реалізована за допомогою команд щознаходяться в таблиці 3.2.

3 Далі, післяотриманнявсіхдодатковихданнихдодатокструктуруватиїх та на основіцьогобудуватисигнатури.

4 Післяпобудови сигнатур, користувачзможепобачитиїх за допомогоювиведенняостанніх на екран.

5 В кінцідодатокзбареже список сформованих сигнатур в документ для їхподальшоговикористання, аналізучизберігання

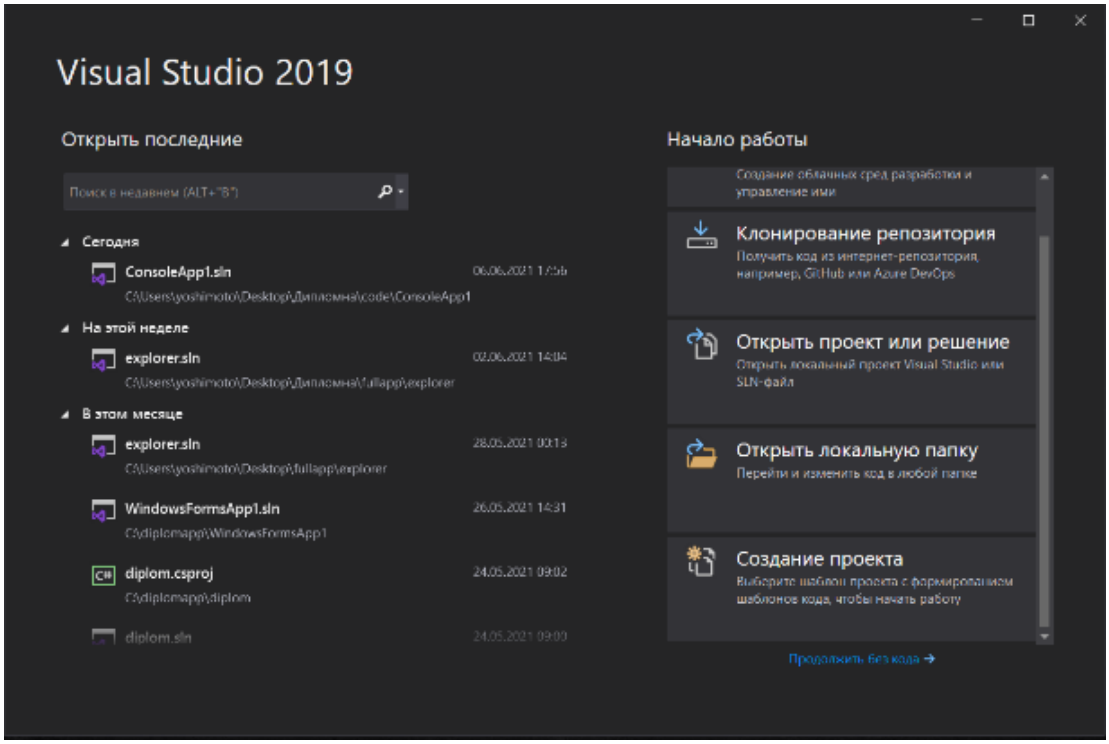
Логічна схема будується для наглядного зображеннявсіхз`єднань, дійдодатку та послідовної / правильноїреалізаціїостаннього. Якщо в простихдодатках схема можезображуватись простою та маленькою, то в великих проектах схема буде великих розмірів для наглядної та послідовноїпобудовидодатку.

Такожлогічна схема дозволяєрозробникам з легкістюінтегровуватирізнімодулі в своїдодатки, що і потребує тема дипломноїроботи ”реалізація модулю формування сигнатур для системиередбаченнявзаємоблокувань”.

Таким чином зрозумілощологічнісхеминеобхідні для коректногоопераннядодатком, і являються одою з найважливішихчастин будь-якого проекту.

Рисунок 3.5 – Логічна схема

Додатокреалізовуємо у виглядіконсольноїкоманди. Для створення проекту запускаємоMicrosoftVisualStudio. Далі у новому вікнівибираємопунк “Створення проекту”, та в наступномувікні тип додатку (Рис.6).



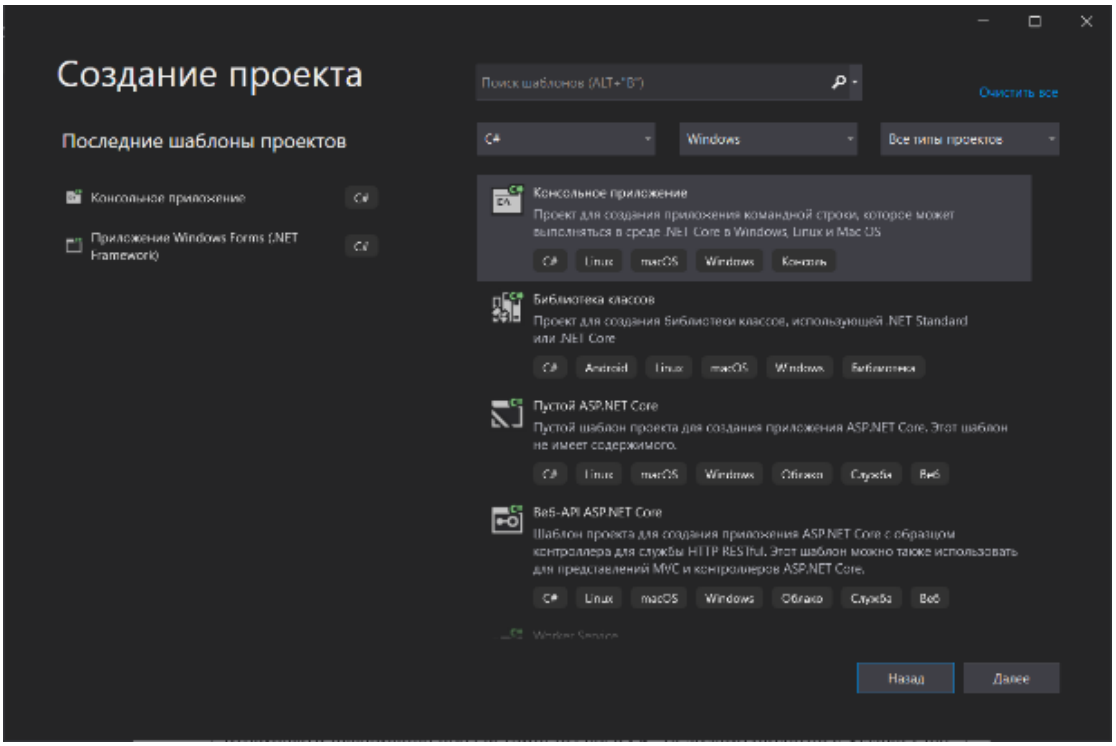


Рисунок 3.6 – Створенняновогопроекту

СпочаткупотрібнопідключитибібліотекуSystem.DiagnosticsщонадаєкласиякідопомагаютьвзаємодіятизсистемнимиресурсамиОС, отримуватиінформаціюпронихтазапотребоюнавітьзмінюватиїї.

ДалівикористаємобібліотекуSystem.Linqщонадаєзмогувикористовуватискороченікомандивиклику, яківвсвоючергузменшуютьрозміркодутароблятьйогобільшгнучкимувлаштуванніта використанні.

Далі автоматично створюєтьсяназвадодатку яку користувачввів при відкриттіVisualStudio(Рис.15)

Післяцьогостворюємо класс Program вякомудодамофункціюvoidMainщо буде виконувати задачу по отриманнюданних з системи.

Спочатку за допомогоюкомандиProcess.GetProcesses().ToList().ForEach [12] отримуємо доступ дисистемноїінформації, а самедомтуп до процесівопераційноїсистеми та за допомогою команд щоописувались у таблиці 3.2 будуємо з них сигнатуру. Для виводуінформації на екранвикористаємо команду Console.Writeline

В кінцізбережимоотриманідані в текстовий документ для подальшогоїхвикористання за допомогоюфункціїStreamWriter

Отжевикориставшивсіцікомандиможливопобудуватинеобхідний модуль формування сигнатур який в подальшомуможливо буде використати для вмонтування в системиередбаченнявзаємоблокуваньабоіншихдодотківщопотребуютьінформацію про процеси.

В результатівиконаннявсіхопераційдодаток буде повертатинекодованісигнатурипроцесів (Рис.25). Такийвигляд сигнатур бувобраний для їхзручногоподальшоговикористання так, як користувач за особистої потреби в подальшомузможевласноручїхзакодовувати в двійковучишіснадцятковусисткму.

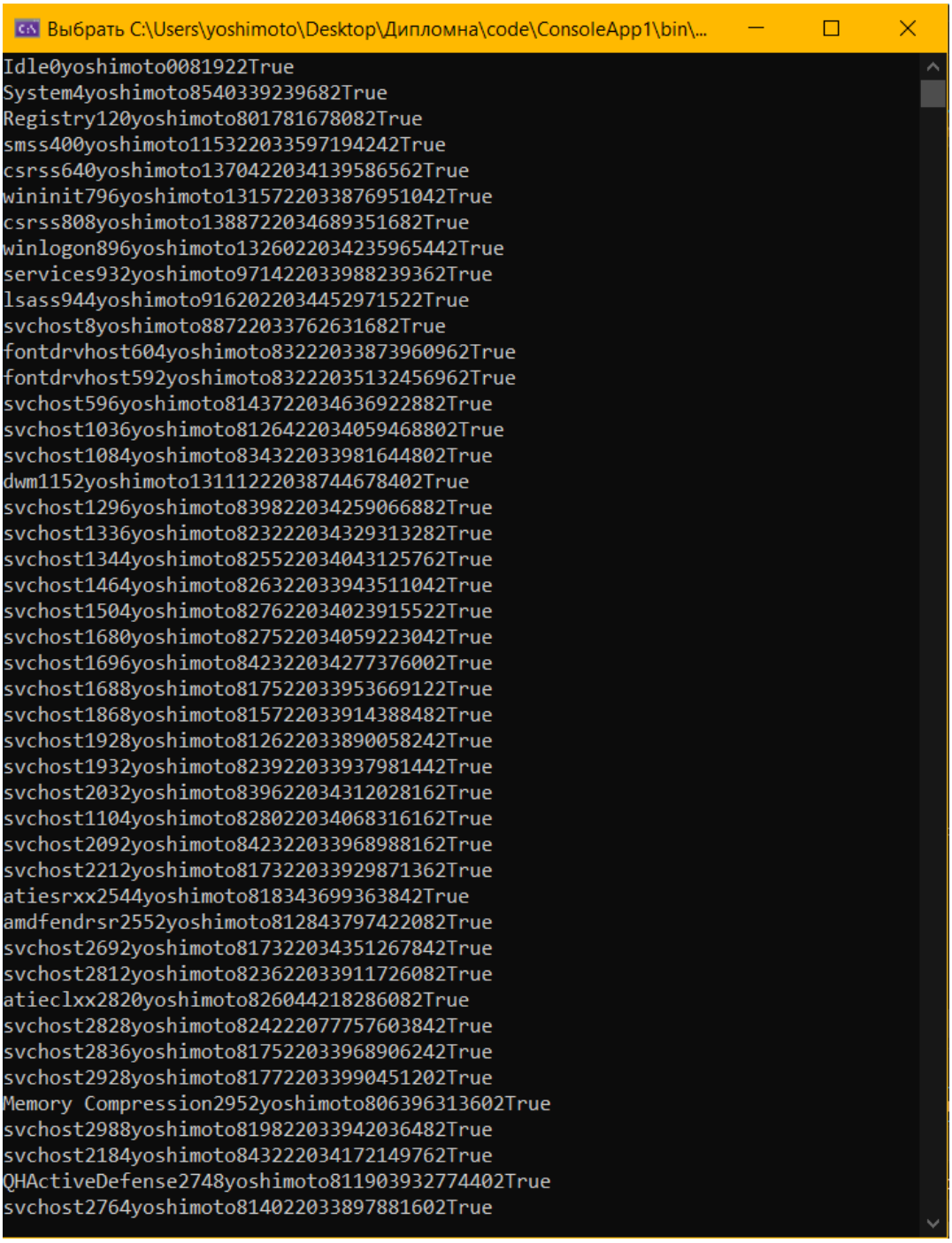


Рисунок 3.7– Приклад роботи модулю

Основними причинами подальшогокодуванняможуть бути потреби для записуінформації в файл чиперенесеннямiжпристроями, а такожзбереженнябезпеки про свій ОС.

У таблиці 3.3 зображеноприкладикодування сигнатур процесівщобулиотримані за допомогоюдодатку.

Таблиця 3.3 – Прикладикодування сигнатур.

Час створення	9:33	9:33	9:33	9:33
Сигнатура процесу	001011010101001 001011100010011 000101100001011 011010011010010 001000011110 000110110001100 101100010010110 000101110001010 001010100100101 0110010110000 101110101011000 001000010010000 100011100000111 000001110100011 0110001101000 011011000111110 010001000011110 000110010001110 100011011001111 0101011011010 1111001001110	2D525C4C5 85B4D221E 1B1962585 C515256585 D5821211C 1C1D1B1A 1B1F221E1 91D1B3D5 B 5E4E	0101100001011001010 0111001011011010010 1000011010000110010 001110100100001 0001100101100010010 1100001011100010100 0101010010010101100 1011000010111010 1011000000110100001 1001000110100001100 1000111010001101100 0110110001110000 0110010010000100011 1000010000000011010 0010000100100000000 1110000100000001 0001000011011000110 1100111101010110110 101111001001110	58594E5B4A 1A191D2119 62585C51525 6585D581A1 91A191D1B1 B1C19211C2 01A21201C2 0 221B1B3D5B 5E4E
Система числення	двійкова	шіснадцяткова	двійкова	шіснадцяткова
Назвапроцесу	Discord.exe *32 (Месенджер для спілкування)		Opera.exe (Веб-браузер)	

3.3 Висновок третього розділу

У третьому розділі було досліджено середовище .NET та його додаткові дані. Виходячи з інформації про середовище вон обуло обране для реалізації поставленного завдання.

При реалізації модулю формування сигнатур, спочатку булла розроблена логічна схема, а вже потім було реалізовано код додатку. Для модулю використовувались такі бібліотеки як:

1. System.
2. System.Diagnostics.
3. System.LinQ
4. System.ComponentModel.
5. Sistem.IO.

Завдяки даним бібліотекам модуль вийшов малих розмірів та інтуітивно зручний у використанні.

В кінці було отримано атуальні сигнатури які можливо закодувати в подальшому для іх використання чи безпеки.

ВИСНОВКИ

В ходіроботибуло проведено детальнийаналізопераційних систем та їхпроцесів. Результатом цьогодосліджено проблему причин виникненнявзаємногоблокування при співпраціпроцесів.

Для вирішенняцієїпроблемибуло проведено аналізіснуючихрішеньцієїпроблеми та виділеноїхосновніплюси та недоліки, такі як: застарілість, вузьконаправленість, проблемиреалізації, неактуальність.

Далібулорозглянуто процесс вираження та формування сигнатур. Та на основіцьогообраносередовище для реалізації модулю. В подальшомудослідженометодиотриманняінформації про процесиопераційноїсистеми, та їхструктуризацію для подальшогоформування сигнатур. За допомогоюцьогорозробленологічну схему модулю та реалізованозабезпечення у виглядідодатку для операційноїсистемиWindows для формування сигнатур, та описано можливікодування та їхвикористання.

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАНЬ

1 Базові поняття технології .NET Framework. URL: https://www.bestprog.net/uk/2016/12/20/базові-поняття-технології-net-framework. (дата звернення: 15.05.21)

2 NET Framework.URL: https://habr.com/ (дата звернення: 15.05.21)

3 Таненбаум С. Современныеоперационныесистемы.СПб.: Питер, 2004. 1040 с.

4 Столлингс В.Операционныесистемы:Издательскийдом "Вильямс", 2002.848 с.

5 Tong Li1, Carla S. Ellis1, Alvin R. Lebeck1, andDaniel J. Sorin: Pulse: A DynamicDeadlockDetectionMechanismUsingSpeculativeExecution. DepartmentofComputerScience, DepartmentofElectricalandComputerEngineering, DukeUniversity. Proceedingsofthe 2005 USENIX AnnualTechnicalConference, Anaheim, California, April 10, 2005.

6 Савенко О.С., Мостовий С.В. Модель прогнозування стану процесів в комп'ютерній системі: радіоелектронні і комп'ютерні системи. Харків: ХАІ, 2008. 109 с.

7 Мостовий С.В. Організація логічного висновку в системі прогнозування стану процесів в персональних комп'ютерах. Збірник праць X міжнародної науково-технічної конференції "Системний аналіз та інформаційні технології" (CAIT-2008). Київ, 2008. 386 с.

8 IntroductionofDeadlockinOperatingSystem.URL: https://www.geeksforgeeks.org/introduction-of-deadlock-in-operating-system/(дата звернення: 16.05.21)

9 Deadlock in Operating System. URL: https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/7_Deadlocks.html (датазвернення: 16.05.21)

10 Deadlock in Operating System | 4 Conditions of Deadlocks | Deadlock Handling - Process Management. URL: https://www.youtube.com/watch?v=UVo9mGARkhQ(дата звернення: 17.05.21)

11 .NET Framework - Compilation Process. URL: https://www.youtube.com/watch?v=6oYcZ-D8Fyw (датазвернення: 19.05.21)

12 Process.GetProcesses Method. URL: https://docs.microsoft.com/uk-ua/dotnet/api/system.diagnostics.process.getprocesses?view=net-5.0 (датазвернення: 17.05.21)

13 Process Class. URL: https://docs.microsoft.com/uk-ua/dotnet/api/system.diagnostics.process?view=net-5.0 (датазвернення: 18.05.21)

14 System.Diagnostics Namespace. URL: https://docs.microsoft.com/uk-ua/dotnet/api/system.diagnostics?view=net-5.0 (датазвернення: 18.05.21)

15 Савенко О.С., Кльоц Ю.П., Мостовий С.В. Дослідження тааналізблокуванняпроцесів в комп'ютернійсистемі.Вісник ХНУ. Хмельницький: ХНУ, 2007. 248-251 с.

16 Роберт Лав. Разработка ядра Linux. СПб.:Вильямс, 2006. 448 с.

17 Стахнов А. Linux. СПб.: BHV-Санкт-Петербург, 2005. 944 с.

18 Роберт Лав. Разработка ядра Linux. СПб.: "Вильямс", 2006 г. 448 с.

19 Робачевский А.М. Операционная система Unix. 2 т. СПб: BHV-Санкт-Петербург, 2007. 656с.

20 Дейтел Х.М., ДейтелП.Дж., Чофнес Д.Р. Операционные системы.1 т: Основы и принципы: Перевод с английского. М: "Бином", 2006. 800с.

21 Платонов Ю. М., Уткин Ю. Г. Диагностика зависания и неисправностей компьютера. Серия "Техномир". Ростов-на-Дону: "Феникс", 2001. 320с.

22 Процессы и домены приложения. URL: https://metanit.com/sharp/tutorial/18.1.php (дата звернення: 19.05.21)

23 Процессы Windows. URL: https://professorweb.ru/my/csharp/assembly/level3/3_1.php(дата звернення: 19.05.21)

24 WindowsIdentity.GetCurrentМетод. URL: https://docs.microsoft.com/ru-ru/dotnet/api/system.security.principal.windowsidentity.getcurrent?view=net-

5.0#System_Security_Principal_WindowsIdentity_GetCurrent (дата звернення: 21.05.21)

25 Adam Nathan. NET and COM: The Complete Interoperability Guide.New-York. 22.01.2002. 50 с.

26 Welcome to "Complete Guide to Asp.Net Core (.NET 5) Web API". URL: .NET documentation. URL: https://docs.microsoft.com/en-us/dotnet/ (датазвернення: 21.05.21)

27 Brachman R. J., Schmoize J. G. An overview of the KL-ONE knowledge representation system. Cognitive Science, 9, 1985p. 171 с.

28 Goldberg A., Robson D. Smalltalk-80: The Language and its Implementation. Reading, MA: Addison-Wesley. 1983p. 74с.

29 Giarratano J. and Riley G. Expert Systems: Principles and Programming, 2nd edn. Boston, MA: PWS Publishing, 1994p. 447с.

30 Shaw M. L. G., PLANET: some experience in creating an integrated system for repertory grid application on a microcomputers. 111 International Journal of Man-Machine Studies. Vol. 17, No. 3. 1982p. 345 с.

31 Уроки C# .NETWindowsForms / #1 - Создание приложения на C# с SQL (базами данных). URL: https://www.youtube.com/watch?v=gp2rA0rgq_0.(дата звернення: 22.05.21)

32 VC#. Как написать диспетчер задач? Управляем процессами на C# в Windows. Урок 33. URL: https://www.youtube.com/watch?v=mcinIXEnZts&t=426s(дата звернення: 22.05.21)

33 Вилле К. Представляем C#. М.: ДМК Пресс, 2001р.

34 Шилдт Г. Полный справочник по C#. Издательский дом «Вильямс», 2007. 752 с.

35 CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# Джеффри Рихтер, 2018р.

36 Исследование операций. 10-е издание - Хемди А. Таха, 2019 р.

37 Хемди А. Таха. Исследование операций. 7-е издание. 2012 р.

38 Гультяев А.К. Управление проектами.КОРОНА принт. 2003 р.

39 ТрофимовС.А. - Rational XDE для Visual Studio .NET. 2004 р.

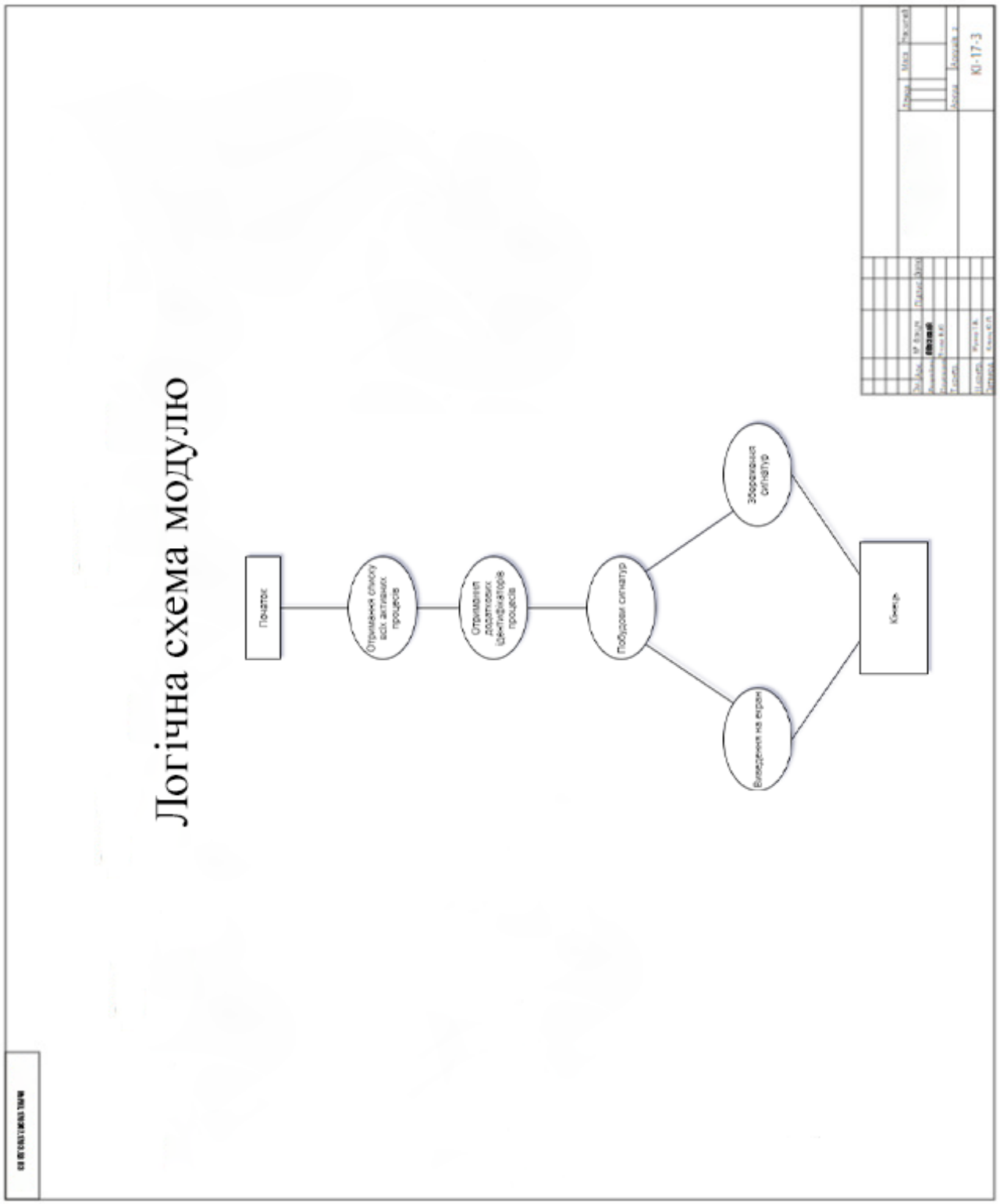
40 Эндрю Троелсен, Филипп Джепик. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд. 2019 р.

41 Jon Galloway, Brad Wilson, K. Scott Allen, David Matson. Professional ASP.NET MVC 5, 2014 р.

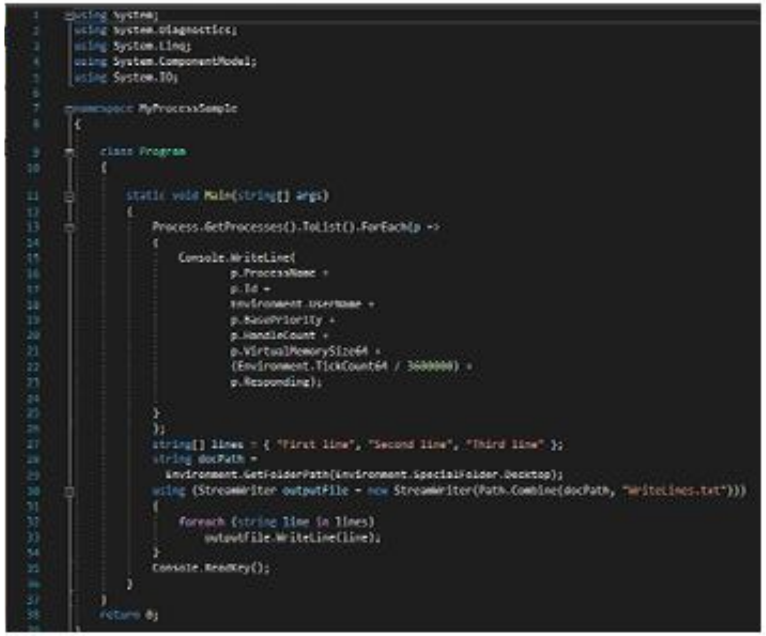
Додаток А

(обовязковий)

Копія графічної частини



Код в компіляторі Visual Studio

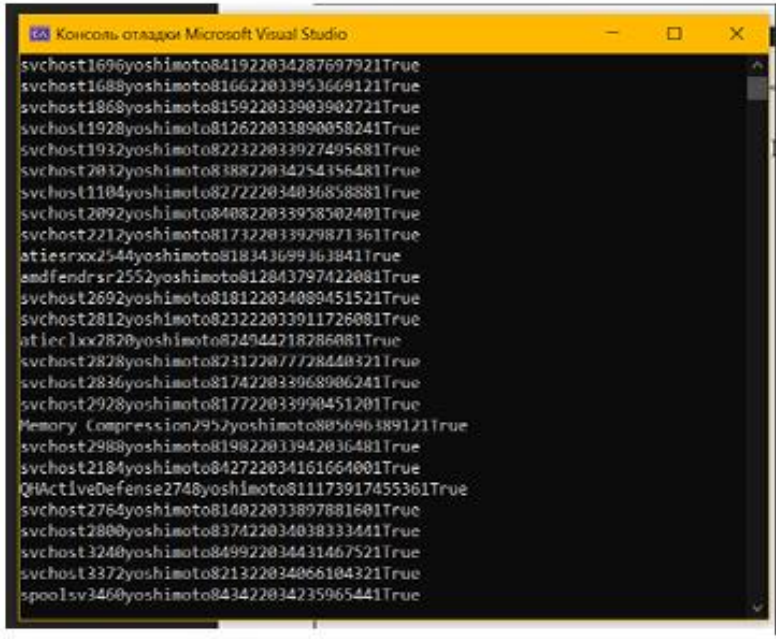


```
using System;
using System.Diagnostics;
using System.Linq;
using System.ComponentModel;
using System.IO;

namespace MyProcessSample
{

    class Program
    {

        static void Main(string[] args)
        {
            Process.GetProcesses().ToList().ForEach(p =>
            {
                Console.WriteLine(
                        p.ProcessName +
                        p.Id +
                        Environment.UserName +
                        p.BasePriority +
                        p.HandleCount +
                        p.VirtualMemorySize64 +
                        (Environment.TickCount64 / 3600000) +
                        p.Responding);

            }
            );
            string[] lines = { "First line", "Second line", "Third line" };
            string docPath =
              Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
            using (StreamWriter outputFile = new StreamWriter(Path.Combine(docPath, "WriteLines.txt")))
            {
                foreach (string line in lines)
                    outputFile.WriteLine(line);
            }
            Console.ReadKey();
        }
    }
    return 0;
}
```

Схеми розробленого Алгоритму

КІ-17-3

Робоче вікно модулю



Зм.	Арк.	№ докум.	Підпис	Дата	
Розроб.		Ковтун О.О.			
Перевір.		Клим В.Ю.			
Т.контр.					
Н.контр.		Жуков Г.В.			
Затверд.		Клим Ю.П.			

Схеми розробленого Алгоритму

Стадія	Маса	Масштаб
Аркуш	Аркушів: 0	

КІ-17-3

Додаток Б

Лістинг коду розробленої системи

```
using System;
usingSystem.Diagnostics;
usingSystem.Linq;
usingSystem.ComponentModel;
using System.IO;

namespaceMyProcessSample
{
classProgram
    {
staticvoidMain(string[] args)
        {
Process.GetProcesses().ToList().ForEach(p =>
            {
Console.WriteLine(
p.ProcessName +
p.Id +
Environment.UserName +
p.BasePriority +
p.HandleCount +
                        p.VirtualMemorySize64 +
                        (Environment.TickCount64 / 3600000) +
p.Responding);

            }
            );
string[] lines = { "First line", "Second line", "Thirdline" };
stringdocPath =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
using (StreamWriteroutputFile = newStreamWriter(Path.Combine(docPath, "WriteLines.txt")))
            {
foreach (stringlineinlines)
outputFile.WriteLine(line);
            }
Console.ReadKey();
        }
    }
return 0;

                                                        }
```

Побудованісигнатури:

Idle0yoshimoto0081926True

System4yoshimoto8556539239686True

Registry120yoshimoto801854873606True

smss404yoshimoto115322033597153286True

SecurityHealthSystray8604yoshimoto815722034275368966True

RtkAudUService648636yoshimoto838244324945926True

QALockHandler8660yoshimoto815643678310406True

WavesSvc648700yoshimoto853444462858246True

unsecapp8836yoshimoto813122033870438406True

SecurityHealthService8876yoshimoto841122034358067206True

SearchUI7880yoshimoto810982238652817 4086True

RadeonSoftware4980yoshimoto8237055526932486True

QHSafeTray9316yoshimoto89373663667206True

svchost9552yoshimoto856122035494871046True

svchost8688yoshimoto829822034223882246True

Discord8164yoshimoto88474324925446True

AMDRSServ10080yoshimoto847348643522566True

Discord9368yoshimoto82552650685446True

Discord6668yoshimoto106926452183046True

Discord2052yoshimoto84512848931846True

Discord9836yoshimoto410118917893126True

Discord7772yoshimoto83222852577286True

SgrmBroker8528yoshimoto89022033772871686True

GoogleCrashHandler2440yoshimoto4183502538246True

GoogleCrashHandler6410192yoshimoto416243853086726True

amdfendrsr1972yoshimoto812943797544966True

atiesrxx1980yoshimoto818443699486726True

svchost2040yoshimoto822322033927577606True

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ КОМП'ЮТЕРНОЇ [illegible]

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студент Обозовий Олексій Вадимович

Тема Модуль формування сигнатур для системи прогнозування взаємоблокувань

Спеціальність 123 – Комп'ютерна інженерія

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:

кількість листів креслень 3; кількість сторінок записки 62.

1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі розроблено модуль формування сигнатур для системи прогнозування взаємоблокувань.

2. Висновок про відповідність кваліфікаційної роботи завданню Кваліфікаційна робота у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі подана загальна характеристика поставленої задачі, сформульована актуальність. Визначені задачі, які необхідно вирішити для досягнення поставленої мети, практична цінність отриманих результатів. У першому розділі проведено огляд операційних систем, процесів, взаємоблокувань та основних програм для їх моніторингу, виконане обгрунтування актуальності теми дослідження і виконана постановка задачі. В другому розділі проведено обгрунтування обраного методу рішення та описано сигнатуру процесу. В третьому розділі розроблено алгоритми та програмне забезпечення побудови сигнатур для системи прогнозування взаємоблокувань.

4. Позитивні сторони роботи Кваліфікаційна робота має комплексну практичну цінність. Практична цінність результатів кваліфікаційної роботи полягає у створені модуля формування сигнатур для системи прогнозування взаємоблокувань, яку можна застосовувати в сучасних ОС для підвищення їх продуктивності та надійності

5. Негативні сторони роботи Розроблений в роботі модуль має вузький функціонал

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконане відповідно до теми кваліфікаційної роботи з дотриманням стандартів. В загальному графічне оформлення виконане якісно, пояснювальна записка відповідає нормам щодо її оформлення.

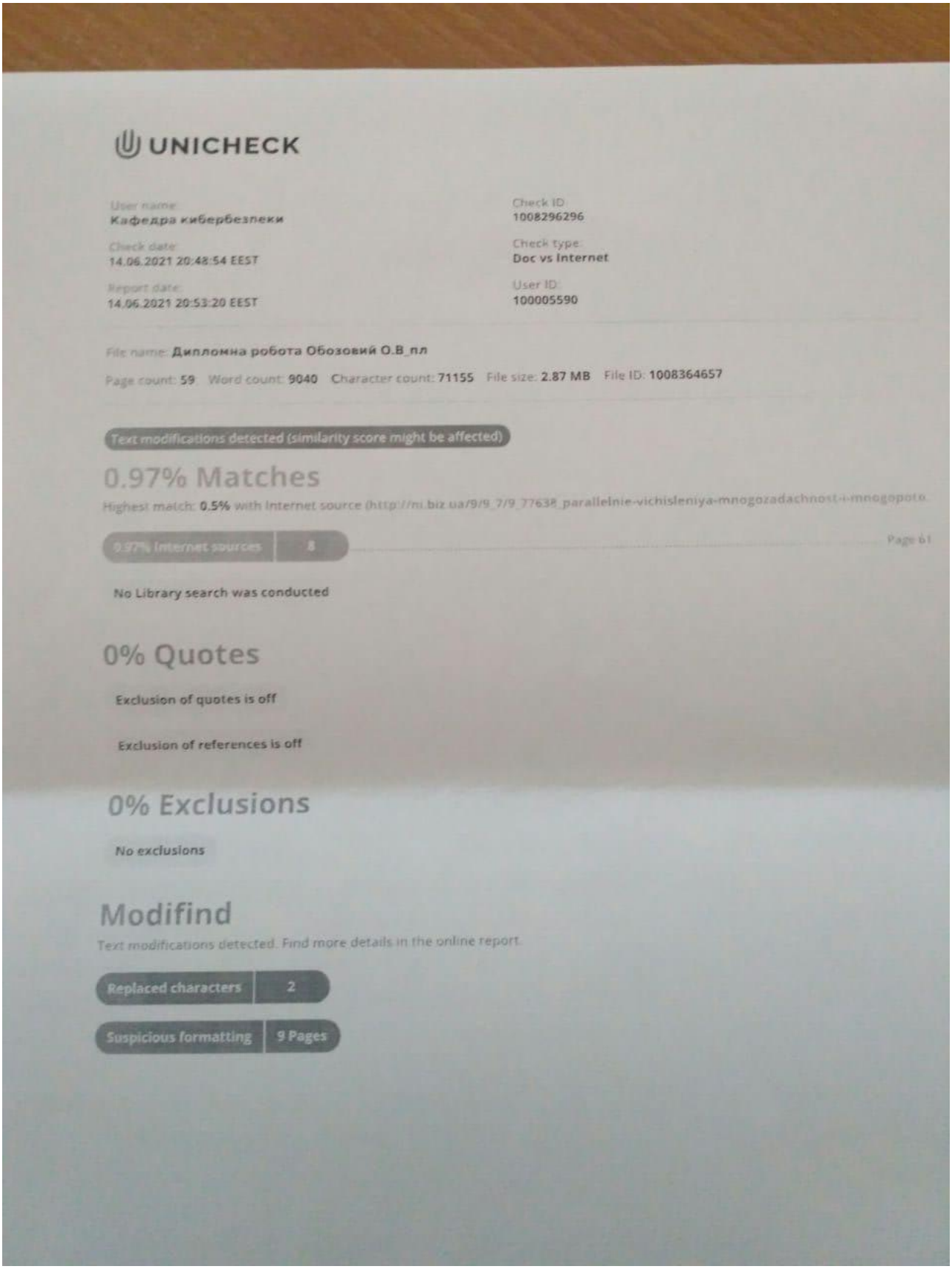
7. Відгук про роботу в цілому В загальному кваліфікаційна робота заслуговує позитивної оцінки. Весь матеріал кваліфікаційної роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої мети.

8. Інші зауваження Окремі описи в пояснювальній записці подано занадто деталізовано, що ускладнює сприйняття матеріалу фахівцями в обраній предметній галузі.

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку «добре» С.

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи) Бездомний І.Р. зав. кафедри ІТТ

«18» 06 2021. (підпис)



UNICHECK

User name:
Кафедра кибербезпеки

Check date:
14.06.2021 20:48:54 EEST

Report date:
14.06.2021 20:53:20 EEST

Check ID:
1008296296

Check type:
Doc vs Internet

User ID:
100005590

File name: Дипломна робота Обозовий О.В_пл

Page count: 59 Word count: 9040 Character count: 71155 File size: 2.87 MB File ID: 1008364657

Text modifications detected (similarity score might be affected)

0.97% Matches

Highest match: 0.5% with Internet source (http://ni.biz.ua/9/9_7/9_77638_parallelnie-vichisleniya-mnogozadachnost-i-mnogopoto...

0.97% Internet sources 8 Page 61

No Library search was conducted

0% Quotes

Exclusion of quotes is off

Exclusion of references is off

0% Exclusions

No exclusions

Modifind

Text modifications detected. Find more details in the online report.

Replaced characters 2

Suspicious formatting 9 Pages

Mon Jun 14 19:59:25 EEST 2021, Муляра І.В, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 0.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 7%**

ID: 93795 Название: Модуль формування сигнатур для системи прогнозування взаємоблоквань Добавлено в БД: 2021-06-14 Авторы: Обозовий Олексій Вадимович Руководители: Мостовий Сергій Володимирович Консультанты: Опоненты:	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
	55993	485	159 (0%)	4 (1%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

Завідувачу кафедри КБКСМ
к.т.н, доцент Кльоц Ю.П.

Обозовий О.В.
ПІБ здобувача вищої освіти

ФПКТС, 4 курсу, групи КІ-17-3

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність плагіату ознайомлений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

18.06.21
дата

підпис

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Синтез та моделювання операційного автомату на основі автомату Мура

Автор: Обозовий Олексій Вадимович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Мостовий Сергій Володимирович

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформлені посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10–40 джерелами на один фрагмент речення;
4) в якості запозичень в окремих місцях системою зафіксовано послідовності чотирьохрозрядних двійкових кодів, які є вхідними даними до великої кількості задач і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості, складає 0.97% що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ С.В. Мостовий

Завідувач кафедри КБКСМ _____ Ю.П Кльоц