МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ОЗНАЙОМЛЕННЯ ТА КЕРУВАННЯ ПРОЦЕСАМИ В ОПЕРАЦІЙНИХ СИСТЕМАХ ДЛЯ ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА. WINDOWS.

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи №1
з дисципліни «Операційні системи»
для студентів першого (бакалаврського) рівня вищої освіти
спеціальності 121 «Інженерія програмного забезпечення»

Затверджено
на засіданні кафедри
програмного забезпечення.
Протокол № __ від __.__.2020 р.

Львів – 2020

Ознайомлення та керування процесами в операційних системах для персонального комп’ютера. Windows.: Методичні вказівки до лабораторної роботи №1 з дисципліни «Операційні системи» для студентів першого (бакалаврського) рівня вищої освіти спеціальності 121 – Інженерія програмного забезпечення / Укл.: В.С. Яковина, О.Д. Грицай, В.Ю. Майхер, Н.О. Семенишин – Львів :, 2020. – 25 с.

Укладачі: Яковина В.С., д-р. техн. наук, проф.

Грицай О.Д., канд. фіз.-мат. наук,

Майхер В.Ю., канд. техн. наук,

Семенишин Н.О.

Відповідальний за випуск Федасюк Д.В., д-р. техн. наук, проф.

Рецензенти Сенів М.М. канд. техн. наук, доц.,

Фечан А.В. д-р. техн. наук, проф.

ЛАБОРАТОРНА РОБОТА №1

Тема. Ознайомлення та керування процесами в операційних системах для персонального комп’ютера. Windows.

Мета. Ознайомитися з процесами та потоками в операційній системі Windows. Навчитися працювати із системними утилітами, що дають можливість отримувати інформацію про процеси, потоки, використовувану ними пам'ять, та іншу необхідну інформацію.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Термін “Операційна система” охоплює багато визначень та функцій. Частково, це через велику різноманітність комп'ютерних систем, які нероздільно пов'язані з операційними системами. Основною метою комп'ютерних систем є виконання програм та полегшення вирішення проблем користувачів. Комп'ютерна техніка побудована для досягнення цієї мети. Оскільки саме обладнання не є простим у використанні, розробляються прикладні програми. Ці програми вимагають певних типових операцій, таких як керування пристроями вводу-виводу. Потім загальні функції контролю та розподілу ресурсів об’єднуються в одне програмне забезпечення: операційну систему. Отже, першим визначенням операційної системи можна вважати:

Операційна система - це сукупність програм, які призначені для керування ресурсами комп'ютера й обчислювальними процесами, а також для організації взаємодії користувача з апаратурою.

З іншої сторони, немає загального визначення, що є складовою частиною операційної системи. Операційна система може містити ті, чи інші функції, залежно від призначення і виду різних комп'ютерних систем. Тому, найпоширенішим визначенням є:

Операційна система - це програма, яка постійно працює на комп'ютері і, зазвичай, називається ядром.

Поряд з ядром існують ще два типи програм: системні програми, які пов'язані з операційною системою, але не обов'язково є частиною ядра, та

прикладні програми, що включають усі програми, не пов'язані з роботою системи.

Функції операційних систем:

- *Керування та розподіл ресурсів.* **Ресурси** - це логічні й фізичні компоненти комп'ютера: оперативна пам'ять, місце на диску, периферійні пристрої, процесорний час тощо. Операційна система здійснює керування процесами, пам’яттю, файловою системою, дисковим простором, операціями вводу/виводу. Операційна система забезпечує механізми захисту та гарантує безпеку операційної системи та користувачів.
- *Керування обчислювальними процесами*. **Обчислювальним процесом** (або завданням) називається послідовність дій, яка задається програмою. У принципі, функції керування процесами можна було б передати кожній прикладній програмі, але тоді програми були б набагато більшими та складнішими. Тому зручніше мати на комп'ютері одну керуючу програму - операційну систему, послугами якої користуватимуться всі інші програми.
- *Забезпечення взаємодії користувача з апаратурою* через інтерфейс користувача. До складу інтерфейсу користувача входить також набір сервісних програм - утиліт. **Утиліта** - це невелика програма, що виконує конкретну сервісну функцію. Утиліти звільняють користувача від виконання рутинних і часом досить складних операцій.

В межах цієї лабораторної роботи пропонується ознайомитися з однією з функцій операційної системи - керування процесами, включаючи моніторинг процесів.

Процес — одне з найважливіших понять у архітектурі операційних систем та програмуванні. **Процес** — об'єкт операційної системи, контейнер системних ресурсів, призначених для підтримки виконання програми. Коли в середовищі операційної системи запускається прикладна програма, система створює спеціальний об’єкт – процес, – який призначений для підтримки її виконання. Хоча може здатися, що програма й процес – поняття схожі, вони фундаментально відрізняються. **Програма** – це статичний набір команд, а **процес** – контейнер для ресурсів, які використовуються при виконанні екземпляра програми. Системна реалізація та функції процесу в різних операційних системах дещо відмінні. Кожен процес представлений в операційній системі структурою даних - **блоком управління процесом (PCB)**. PCB підтримується для процесу протягом усього його життя і видаляється

після закінчення процесу. PCB зберігає всю інформацію, необхідну для відстеження процесу (рис.1). Архітектура PCB повністю залежить від операційної системи і може містити різну інформацію в різних операційних системах.

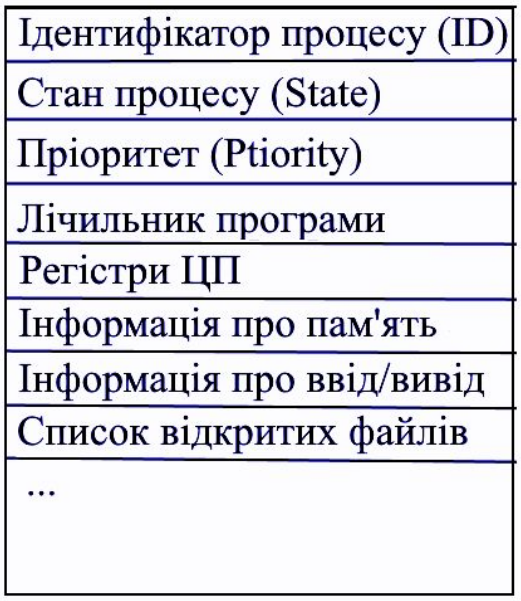


Рис.1. Блок управління процесом

Структура процесу у Windows

На найвищому рівні абстракції процес у Windows містить:

- закритий віртуальний адресний простір – діапазон адрес віртуальної пам'яті, яким може користуватися процес;
- програму, що виконується – початковий код і спроектовані на віртуальний адресний простір процесу дані;
- список відкритих дескрипторів (handles) різних системних ресурсів – комунікаційних портів, файлів та інших об'єктів, доступних усім потокам даного процесу;
- контекст захисту (security context), який називають маркером доступу (access token), – він ідентифікує користувача та групи безпеки й привілеї, зіставлені з процесом;
- унікальний ідентифікатор процесу;
- мінімум один потік.

Отже, **процес** – це абстракція для опису програми, що виконується. Процес є об’єктом операційної системи, що складається з адресного простору пам’яті і набору структур даних, що є у ядрі. **Адресний простір** – сукупність сторінок пам’яті, що були виділенні ядром для виконання процесу. У нього

завантажуються код процесу, бібліотеки функцій, змінні, вміст стеку та інша допоміжна інформація (рис.2).

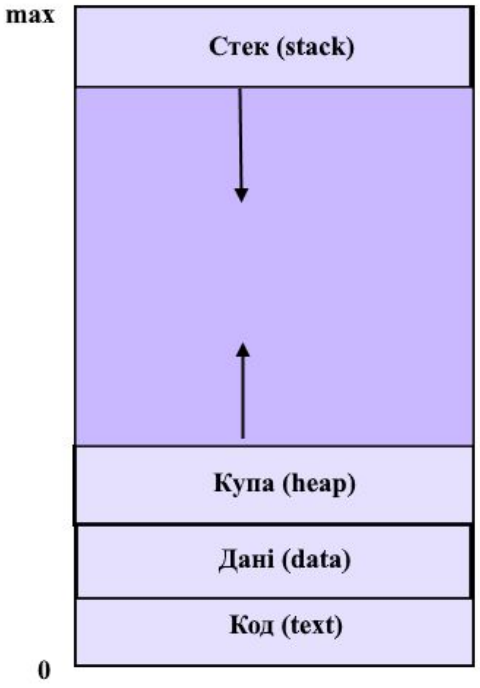


Рис.2 Макет процесу в пам’яті

У структурах даних ядра зберігається інформація про кожний процес.

Загальні характеристики процесу

Дерево процесів.

Кожному процесу мають бути виділені певні системні ресурси, зокрема: процесор, пам'ять, доступ до пристроїв вводу-виводу. Кожен процес має «батька» (батьківський процес). Він також може мати (але не обов'язково) «нащадків» (дочірні процеси). В сукупності вони формують дерево процесів.

Типи процесів

В операційній системі **Windows** процеси можна поділити на:

- ***Процеси рівня користувача*.**
  - Використовують власний адресний простір.
  - Виконуються від імені конкретного користувача.
  - Поділяються на:
    - *системні процеси*
    - *сервіси (Windows Services)*
    - *прикладі програми.*
- ***Процеси рівня ядра*.**
  - Ділять адресний простір.
  - Мають прямий доступ до апаратного забезпечення.

- До них належать Executive Services, Microkernel, HAL.

Пріоритет процесу.

Процес має певний пріоритет, який впливає на кількість процесорного часу, який виділятиметься його потокам.

У операційній системі **Windows** передбачено 32 рівні пріоритету — від 0 до 31. Ці значення групуються так:

- шістнадцять рівнів реального часу (16-31);
- п'ятнадцять динамічних рівнів (1-15);
- один системний рівень (0), зарезервований для потоку обнулення сторінок пам'яті (zero page thread).

У Windows кожен процес має один із таких базових класів пріоритету:

- IDLE_PRIORITY_CLASS,
- BELOW_NORMAL_PRIORITY_CLASS,
- NORMAL_PRIORITY_CLASS,
- ABOVE_NORMAL_PRIORITY_CLASS,
- HIGH_PRIORITY_CLASS,
- REALTIME_PRIORITY_CLASS.

За замовчуванням клас пріоритету процесу набуває значення NORMAL_PRIORITY_CLASS.

Процес як системний об'єкт призначений для "обслуговування" потоків. Сам по собі процес не виконує ніяких дій, для цього призначений потік. Потік - це послідовність машинних команд, які Windows сприймає, як єдине ціле (набір регістрів процесора). Потік має вказівник на команду, що у цей момент виконується, і вказівник на стек, де зберігаються локальні змінні потоку. Потік може створювати інші потоки. Всі потоки, які належать одному процесу, маю доступ до всіх ресурсів цього процесу.

Потік містить такі важливі елементи:

- завантажений для виконання код;
- вміст набору регістрів процесора, що відображають стан процесора;
- два стеки, один із яких використовується потоком при виконанні в режимі ядра, а інший — у користувацькому режимі;
- закриту область пам'яті, яку називають локальною пам'яттю потоку (thread-local storage, TLS); вона використовується підсистемами, бібліотеками виконуваних систем (run-time libraries) і DLL;
- унікальний ідентифікатор потоку;

- іноді потоки мають свій контекст захисту, який використовується багатопотоковими серверними програмами, що підміняють контекст захисту клієнтів.

В межах кожного класу пріоритету можуть бути наступні рівні пріоритету потоків:

THREAD_PRIORITY_IDLE,
THREAD_PRIORITY_LOWEST,
THREAD_PRIORITY_BELOW_NORMAL,
THREAD_PRIORITY_NORMAL,
THREAD_PRIORITY_ABOVE_NORMAL,
THREAD_PRIORITY_HIGHEST,
THREAD_PRIORITY_TIME_CRITICAL.

За замовчуванням потоки створюються із значенням пріоритету THREAD_PRIORITY_NORMAL. Значення рівнів пріоритету потоків THREAD_PRIORITY_ABOVE_NORMAL та THREAD_PRIORITY_HIGHEST найчастіше використовуються у потоках, що призначені для взаємозв'язку з користувачами. Фонові потоки, які не потребують значних процесорних ресурсів, можуть мати пріоритет THREAD_PRIORITY_BELOW_NORMAL або THREAD_PRIORITY_LOWEST. Рівні пріоритету потоку призначаються з урахуванням двох різних точок зору — Windows API і ядра Windows. Windows API спочатку впорядковує процеси за класами пріоритету, призначеними при їхньому створенні (Realtime (реального часу), High (високий), Above Normal (вище звичайного), Normal (звичайний), Below Normal (нижче звичайного) і Idle (простоюючий)), а потім — за відносним пріоритетом індивідуальних потоків у межах цих процесів (Time-critical (критичний за часом), Highest (найвищий), Above Normal (вище звичайного), Normal (звичайний), Below Normal (нижче звичайного), Lowest (найменший) і Idle (простоюючий)). Базовий пріоритет кожного потоку у Windows API встановлюється, виходячи із класу пріоритету його процесу й відносного пріоритету самого потоку.

Робочий цикл процесу

Під час запуску процесу проходять такі стадії: користувач вказує програму, яку потрібно виконати операційна система створює адресний простір для процесу і структури, які описують новий процес заповнюються структури, які описують новий процес з файлу, який містить виконавчий файл, в адресний простір процесу копіюються код і дані встановлюється стан процесу «готовий до виконання» новий процес додається до черги процесів,

які очікують на процесор керування повертається оболонці користувача. Загалом весь робочий цикл процесу, незалежно від операційної системи можна представити станами (рис.3):

- Новий
- Готовий
- Виконується
- Очікує
- Завершився

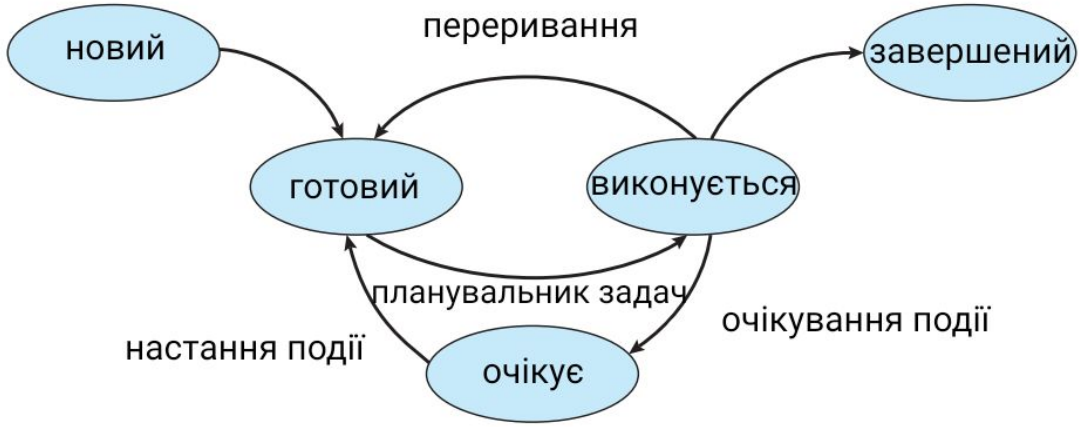


Рис.3 Стани процесів

Для конкретної операційної системи схема станів процесу може бути деталізована.

Найважливішою складовою частиною процесу є потік.

Стани потоків у Windows та можливі переходи між станами потік може бути в таких станах:

- **Initialized** (ініціалізований). У цей стан потік входить у процесі свого створення.
- **Ready** (готовий). Потік у стані готовності очікує виконання. Вибираючи наступний потік для виконання, диспетчер бере до уваги тільки пул потоків, готових до виконання.
- **Standby** (простоює). Потік у цьому стані вже вибраний наступним для виконання на конкретному процесорі. У підходящий момент диспетчер перемикає контекст на цей потік. У стані Standby може перебувати тільки один потік для кожного процесора в системі. Проте потік може бути витиснений навіть у цьому стані (якщо, наприклад, до початку виконання такого потоку до виконання буде готовий потік з вищим пріоритетом).

- **Running** (виконується). Потік переходить у цей стан і починає виконуватись відразу після того, як диспетчер перемикає на неї контекст. Виконання потоку припиняється, коли він завершується, витискується потоком з вищим пріоритетом, перемикає контекст на інший потік, самостійно переходить у стан очікування або минає виділений йому квант процесорного часу (та інший потік з тим же пріоритетом готова до виконання).
- **Waiting** (очікує). Потік входить у цей стан кількома способами. Він може самостійно почати очікування на синхронізуючому об'єкті або його змушує до цього підсистема оточення. Після закінчення очікування, залежно від пріоритету, або негайно починає виконуватися, або переходить у стан Ready.
- **Transition** (перехідний стан). Потік переходить у цей стан, якщо він готовий до виконання, але його стек ядра вивантажений з пам'яті. Щойно цей стек завантажується у пам’ять, потік переходить у стан Ready.
- **Terminated** (завершений). Закінчуючи виконання, потік переходить у стан Terminated. Після цього інформація про потік у виконавчій системі може бути знищена. Можливі переходи між станами потоків показано на діаграмі (рис.4).

Рис.4. Переходи між станами потоків

Моніторинг процесів користувачем

Моніторинг процесів в ос Windows здійснюється через ряд утиліт. Найпоширенішою є вбудована утиліта **Диспетчер Задач** (Task manager). Вбудований диспетчер завдань Windows забезпечує швидкий перелік процесів у системі. Запустити диспетчер завдань можна одним із чотирьох способів:

- Натисніть Ctrl + Shift + Esc.

- Клацніть правою кнопкою миші панель завдань і натисніть кнопку Запустити диспетчер завдань.
- Натисніть Ctrl + Alt + Delete і натисніть кнопку Пуск диспетчера завдань.
- Запустіть виконуваний файл Taskmgr.exe.

Спочатку *Диспетчер задач* запускається в скороченому вигляді. Для отримання повнішої інформації про запущені процеси необхідно натиснути *Більше деталей*. За замовчуванням відкривається вкладка *Процеси*.

На вкладці *Процеси* відображається список процесів із чотирма стовпцями: ЦП, Пам'ять, Диск та Мережа (рис.5). Викликавши контекстне меню на заголовку можна додати більше стовпців (ім’я, ідентифікатор процесу, тип, статус, видавець та командний рядок). Деякі процеси можна додатково розширити, показуючи видимі вікна верхнього рівня, створені процесом.

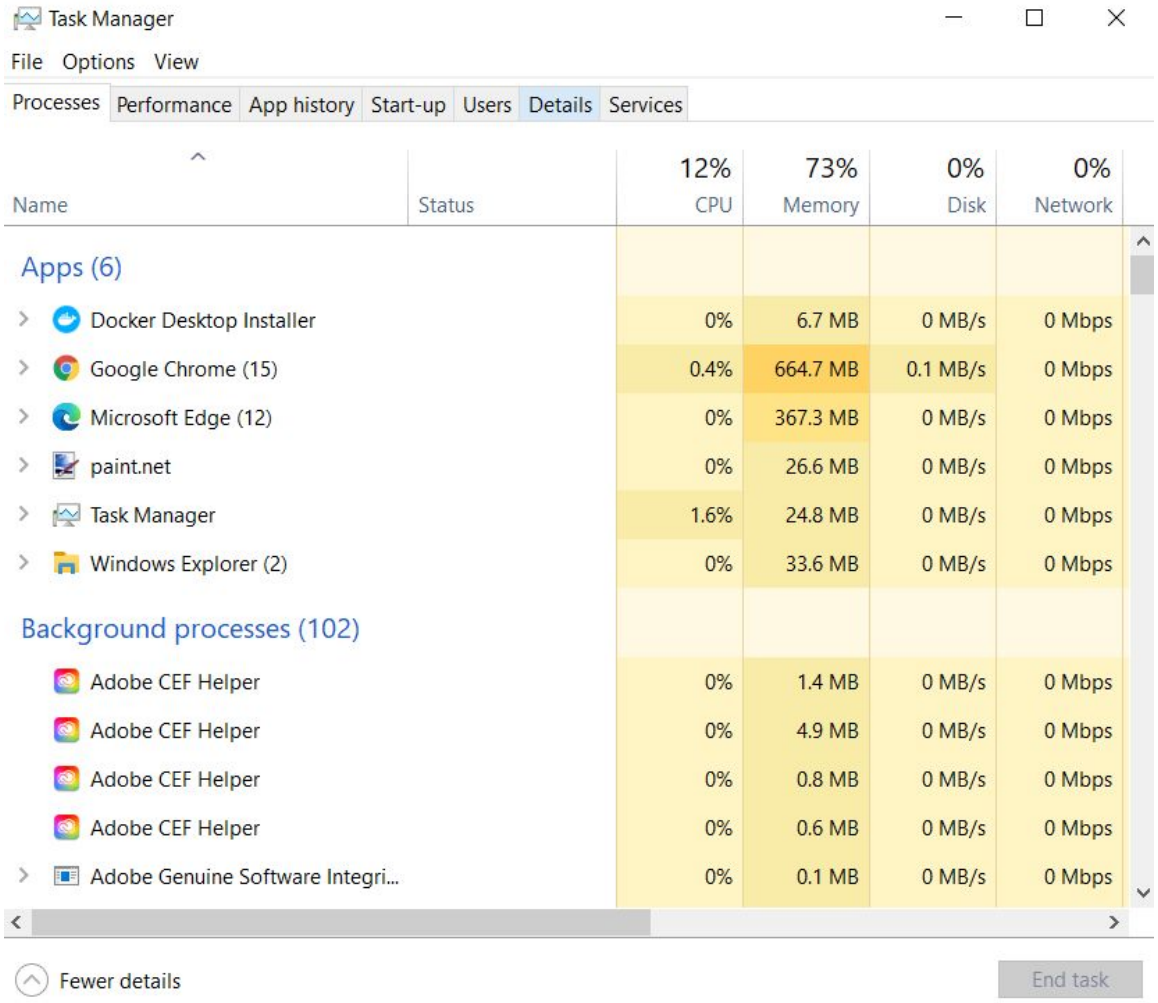
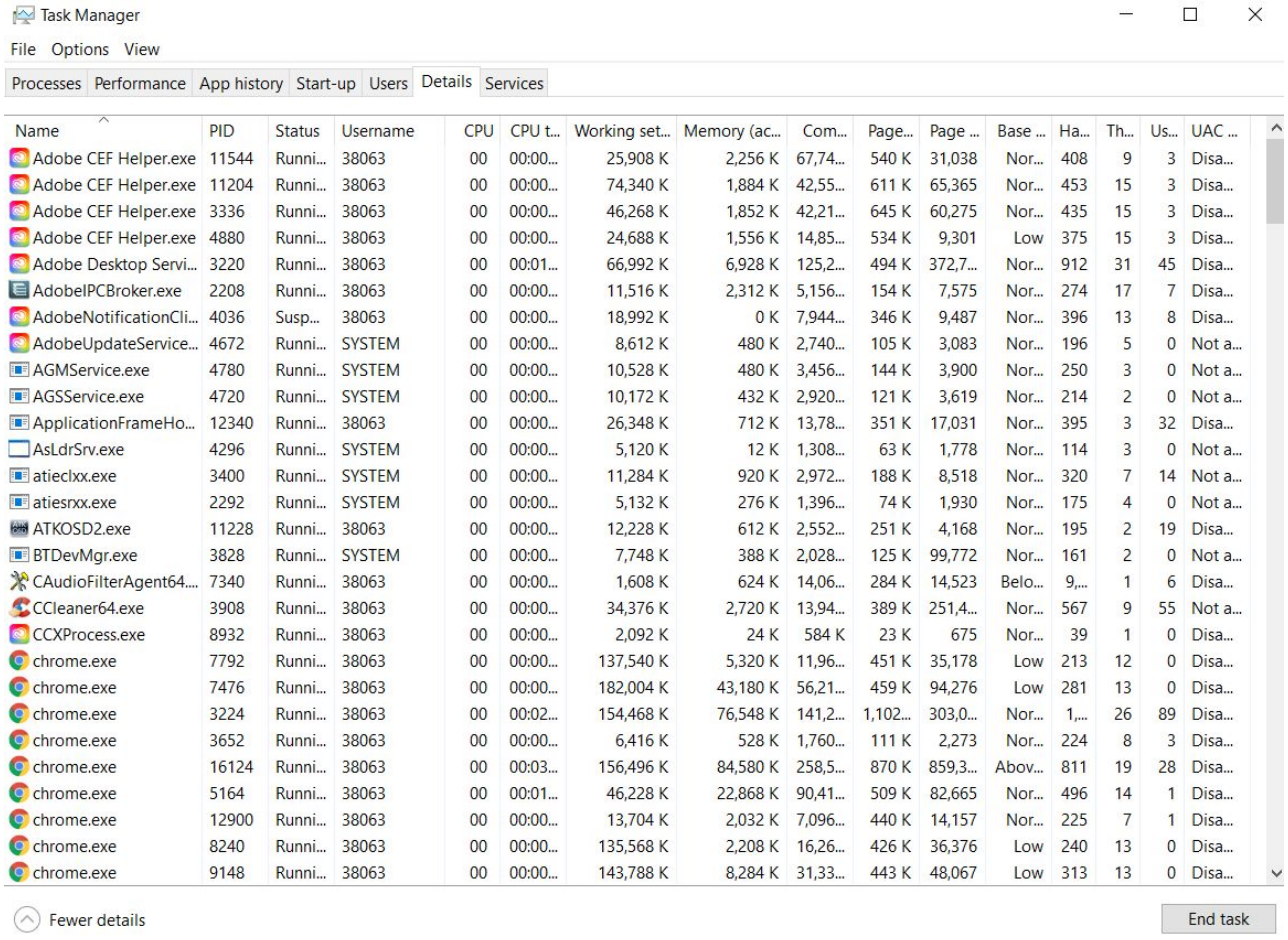


Рис.5 Диспетчер задач. Вкладка *Процеси*

Для отримання детальної інформації про процеси необхідно перейти до вкладки *Деталі* (рис.6). На вкладці *Деталі* також відображаються процеси, але це робиться більш компактно. Вкладка *Деталі* не відображає вікна, створені процесами, але забезпечує більш різноманітні інформаційні стовпці. Процеси ідентифікуються за назвою образу (програми), екземпляром якого вони є. На відміну від деяких об'єктів у Windows, процесам не можна присвоювати глобальні імена.



Name	PID	Status	Username	CPU	CPU t...	Working set...	Memory (ac...	Com...	Page...	Page ...	Base ...	Ha...	Th...	Us...	UAC ...
Adobe CEF Helper.exe	11544	Runni...	38063	00	00:00...	25,908 K	2,256 K	67,74...	540 K	31,038	Nor...	408	9	3	Disa...
Adobe CEF Helper.exe	11204	Runni...	38063	00	00:00...	74,340 K	1,884 K	42,55...	611 K	65,365	Nor...	453	15	3	Disa...
Adobe CEF Helper.exe	3336	Runni...	38063	00	00:00...	46,268 K	1,852 K	42,21...	645 K	60,275	Nor...	435	15	3	Disa...
Adobe CEF Helper.exe	4880	Runni...	38063	00	00:00...	24,688 K	1,556 K	14,85...	534 K	9,301	Low	375	15	3	Disa...
Adobe Desktop Servi...	3220	Runni...	38063	00	00:01...	66,992 K	6,928 K	125,2...	494 K	372,7...	Nor...	912	31	45	Disa...
AdobeIPCBroker.exe	2208	Runni...	38063	00	00:00...	11,516 K	2,312 K	5,156...	154 K	7,575	Nor...	274	17	7	Disa...
AdobeNotificationCli...	4036	Susp...	38063	00	00:00...	18,992 K	0 K	7,944...	346 K	9,487	Nor...	396	13	8	Disa...
AdobeUpdateService...	4672	Runni...	SYSTEM	00	00:00...	8,612 K	480 K	2,740...	105 K	3,083	Nor...	196	5	0	Not a...
AGMService.exe	4780	Runni...	SYSTEM	00	00:00...	10,528 K	480 K	3,456...	144 K	3,900	Nor...	250	3	0	Not a...
AGSService.exe	4720	Runni...	SYSTEM	00	00:00...	10,172 K	432 K	2,920...	121 K	3,619	Nor...	214	2	0	Not a...
ApplicationFrameHo...	12340	Runni...	38063	00	00:00...	26,348 K	712 K	13,78...	351 K	17,031	Nor...	395	3	32	Disa...
AsLdrSrv.exe	4296	Runni...	SYSTEM	00	00:00...	5,120 K	12 K	1,308...	63 K	1,778	Nor...	114	3	0	Not a...
atieclxx.exe	3400	Runni...	SYSTEM	00	00:00...	11,284 K	920 K	2,972...	188 K	8,518	Nor...	320	7	14	Not a...
atiesrxx.exe	2292	Runni...	SYSTEM	00	00:00...	5,132 K	276 K	1,396...	74 K	1,930	Nor...	175	4	0	Not a...
ATKOSD2.exe	11228	Runni...	38063	00	00:00...	12,228 K	612 K	2,552...	251 K	4,168	Nor...	195	2	19	Disa...
BTDevMgr.exe	3828	Runni...	SYSTEM	00	00:00...	7,748 K	388 K	2,028...	125 K	99,772	Nor...	161	2	0	Not a...
CAudioFilterAgent64....	7340	Runni...	38063	00	00:00...	1,608 K	624 K	14,06...	284 K	14,523	Belo...	9,...	1	6	Disa...
CCleaner64.exe	3908	Runni...	38063	00	00:00...	34,376 K	2,720 K	13,94...	389 K	251,4...	Nor...	567	9	55	Not a...
CCXProcess.exe	8932	Runni...	38063	00	00:00...	2,092 K	24 K	584 K	23 K	675	Nor...	39	1	0	Disa...
chrome.exe	7792	Runni...	38063	00	00:00...	137,540 K	5,320 K	11,96...	451 K	35,178	Low	213	12	0	Disa...
chrome.exe	7476	Runni...	38063	00	00:00...	182,004 K	43,180 K	56,21...	459 K	94,276	Low	281	13	0	Disa...
chrome.exe	3224	Runni...	38063	00	00:02...	154,468 K	76,548 K	141,2...	1,102...	303,0...	Nor...	1,...	26	89	Disa...
chrome.exe	3652	Runni...	38063	00	00:00...	6,416 K	528 K	1,760...	111 K	2,273	Nor...	224	8	3	Disa...
chrome.exe	16124	Runni...	38063	00	00:03...	156,496 K	84,580 K	258,5...	870 K	859,3...	Abov...	811	19	28	Disa...
chrome.exe	5164	Runni...	38063	00	00:01...	46,228 K	22,868 K	90,41...	509 K	82,665	Nor...	496	14	1	Disa...
chrome.exe	12900	Runni...	38063	00	00:00...	13,704 K	2,032 K	7,096...	440 K	14,157	Nor...	225	7	1	Disa...
chrome.exe	8240	Runni...	38063	00	00:00...	135,568 K	2,208 K	16,26...	426 K	36,376	Low	240	13	0	Disa...
chrome.exe	9148	Runni...	38063	00	00:00...	143,788 K	8,284 K	31,33...	443 K	48,067	Low	313	13	0	Disa...

Рис.6 Диспетчер задач. Вкладка *Деталі*

Щоб відобразити додаткові деталі, клацніть правою кнопкою миші рядок заголовка та натисніть *Вибрати стовпці* (рис. 7).

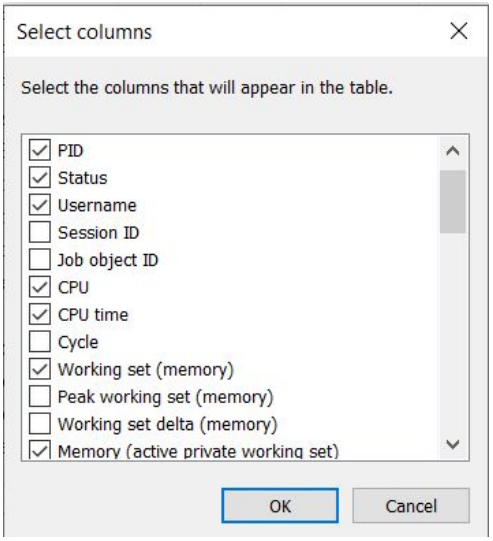


Рис.7 Вибір стовпців на вкладці *Деталі*

Деякі основні стовпці, що надають інформацію про процеси:

PID (Ідентифікатор процесу). Унікальний ідентифікаційний номер, що присвоєний ОС кожному процесу, який допомагає процесору ідентифікувати кожен процес окремо.

Username (Ім'я користувача). Обліковий запис користувача, під яким виконується процес.

Session ID (Ідентифікатор сеансу). Використовується для ідентифікації власника процесу в системі з декількома користувачами, кожний користувач має свій унікальний ідентифікатор сеансу.

CPU (Завантаження ЦП). Процент часу, який процес використовував ЦП.

CPU Time (Час ЦП). Загальний час ЦП в секундах, що використовується процесом з моменту його запуску.

Status (Статус). Стовпець *Статус* відображає стан процесу. Але для різних типів процесів це значення можна по-різному трактувати. Для процесів, які не мають жодного користувальницького інтерфейсу, стан *Running* має бути звичайним випадком, хоча всі потоки можуть чекати чогось, наприклад, об'єкта ядра або завершення якоїсь операції вводу-виводу. Інший варіант стану таких процесів - *Suspended*, коли всі потоки в процесі перебувають у призупиненому стані. (Це може бути досягнуто програмно, викликаючи недокументований власний API NtSuspendProcess у процесі). Для процесів, що створюють користувальницький інтерфейс, значення стану *Running* означає, що інтерфейс відповідає. Тобто, потік, який створив вікно, чекає введення в інтерфейсі користувача. *Suspended* стан можливий і у випадку, що не стосується користувацького інтерфейсу, але для додатків Windows *Suspended*

зазвичай виникає, коли програма втрачає статус переднього плану, оскільки користувач мінімізує вікно. Такі процеси призупиняються через 5 секунд, вони не споживають ніяких процесорних або мережевих ресурсів, що дозволяє новій програмі переднього плану отримувати всі машинні ресурси. Це особливо важливо для пристроїв, що працюють від акумуляторів, таких як планшети та телефони. Третім можливим значенням стану є *Not Responding*. Це може статися, якщо потік у процесі, який створив користувальницький інтерфейс, не перевіряв свою чергу повідомлень на дії, пов'язані з інтерфейсом користувача, принаймні 5 секунд. Процес (насправді потік, який володіє вікном) може бути зайнятий виконанням якоїсь інтенсивної роботи з процесором або очікуванням чогось іншого (наприклад, завершення операції введення-виведення). У будь-якому випадку, користувальницький інтерфейс зависає, і Windows вказує на це.

Commit size - включає лише сторінки пам'яті, які були виділені. Тобто - це віртуальна пам'ять, «підкріплена» (backed) тільки фізичною пам'яттю або pagefile-ом. Обсяг виділеної пам'яті процесу характеризує фактичний обсяг пам'яті, що використовує процес.

Working Set - це набір сторінок, які процес розмістив у фізичній (RAM) пам'яті. На відміну від *Commit size* даний розмір враховує лише те, що знаходиться в оперативній пам'яті, а також додатково ще й сторінки з відображених у пам'ять файлів (розділювана пам'ять, наприклад динамічні бібліотеки). Working Set процесу ділиться на Shareable і Private. ***Shareable*** - це відображенні у пам'ять файли (в тому числі і pagefile backed), вірніше ті частини, які в даний момент дійсно представлені в адресному просторі процесу фізичної сторінкою, а ***Private*** - це купа, стеки, внутрішні структури даних типу PEB / TEB і т.д. даного процесу (тільки тієї частини купи та інших структур, які фізично знаходяться в адресному просторі процесу), які не можуть бути розділені з іншими процесами.

Peak working set (Піковий робоча множина). Максимальний об'єм пам'яті робочої множини сторінок, що використовується процесом.

Working set delta (Дельта робочої множини). Кількість змін в пам'яті робочої множини.

Paged pool. Пули пам`яті (memory pools) - об`єкти, які створюються і використовуються програмами і операційною системою, зберігаються в так званих пулах пам'яті. Переміщуваний або нерезидентний пул (paged pool) містить об'єкти, які можна при необхідності вивантажити на диск.

NP pool. Непереміщуваний або резидентний пул (non-paged pool) - дані або код, які повинні залишитися в пам`яті і не можуть бути записані або зчитані з диска.

Page faults. Помилки сторінок виникають коли програма звертається до сторінки коду або даних, яка не міститься в робочому наборі. І повинна бути знайдена десь в іншому місці. Програмні переривання (soft page fault) - це коли програма звертається до сторінки, яка знаходиться в пам'яті, але поза робочим набором. В цьому випадку не потрібно відновлення сторінки з диска. Апаратні переривання (hard page fault) - це коли програма звертається до сторінки, яка не міститься у фізичній пам'яті (RAM) і підлягає відновленню з диска. Збої сторінок такого типу найкраще показують наявність вузьких місць в конфігурації пам'яті. Більше 5 збоїв в секунду говорять про необхідність додавання RAM.

Base priority (Базовий пріоритет). Базовий клас пріоритету процесу, що впливає на планування потоків процесу.

Threads (Потоки). У стовпці *Потоки* відображається кількість потоків у кожному процесі. Зазвичай, це число має бути принаймні одиницею, оскільки неможливо створити процес без потоку. Якщо процес показує нульові потоки, це означає, що процес не може бути видалений з якоїсь причини - можливо, через якийсь код помилки драйвера.

Handles (Дескриптори). Стовпець *Дескриптори* показує кількість дескрипторів для об'єктів ядра, відкритих потоками, що виконуються в процесі.

User objects (Об'єкти користувача). Кількість об'єктів користувача, що використовуються в даний момент процесом. Об'єкт USER - це об'єкт з Window Manager, який містить вікна, меню, курсори, значки, монітори та інші внутрішні об'єкти.

GDI objects (Об'єкти GDI). Кількість об'єктів бібліотеки графічних інтерфейсів пристроїв (GDI) прикладного програмування (API) для пристроїв виводу графіки.

З допомогою даної утиліти можна керувати процесами через контекстне меню для кожного процесу:

- змінювати пріоритет (рис. 8),

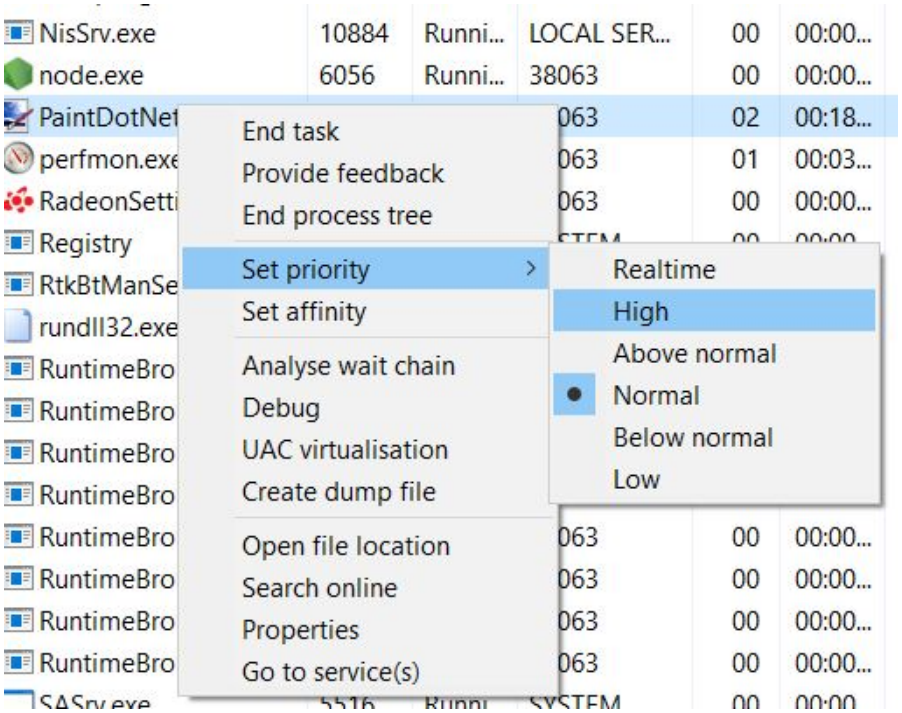


Рис. 8. Зміна пріоритету

- задавати відповідність виконання на певному ядрі (рис. 9),

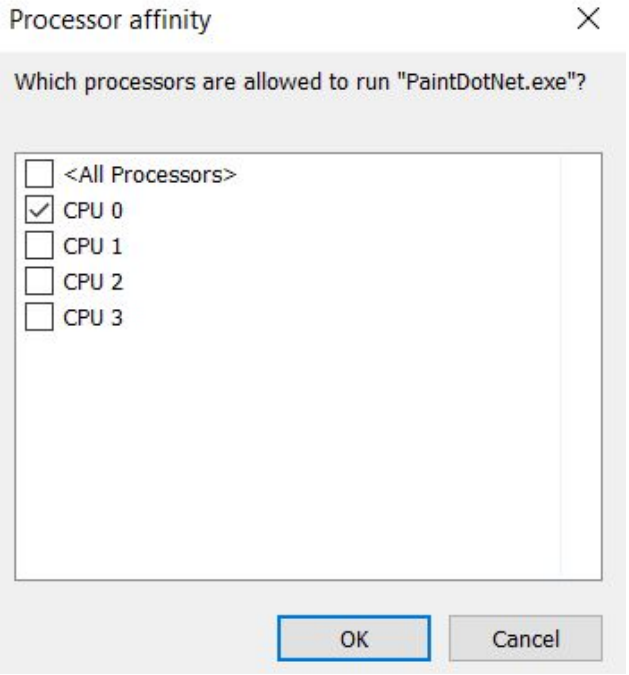


Рис. 9. Задання виконання на певному ядрі

- завершувати (рис. 10).

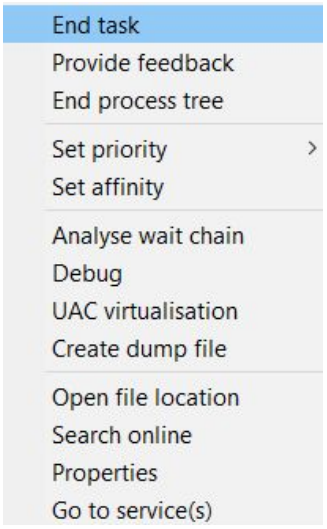


Рис. 10. Завершення роботи процесу

Більші можливості моніторингу і керування процесами надає утиліта **Process Explorer** (рис. 11). Під час запуску **Process Explorer** за замовчуванням відображається дерево процесів.

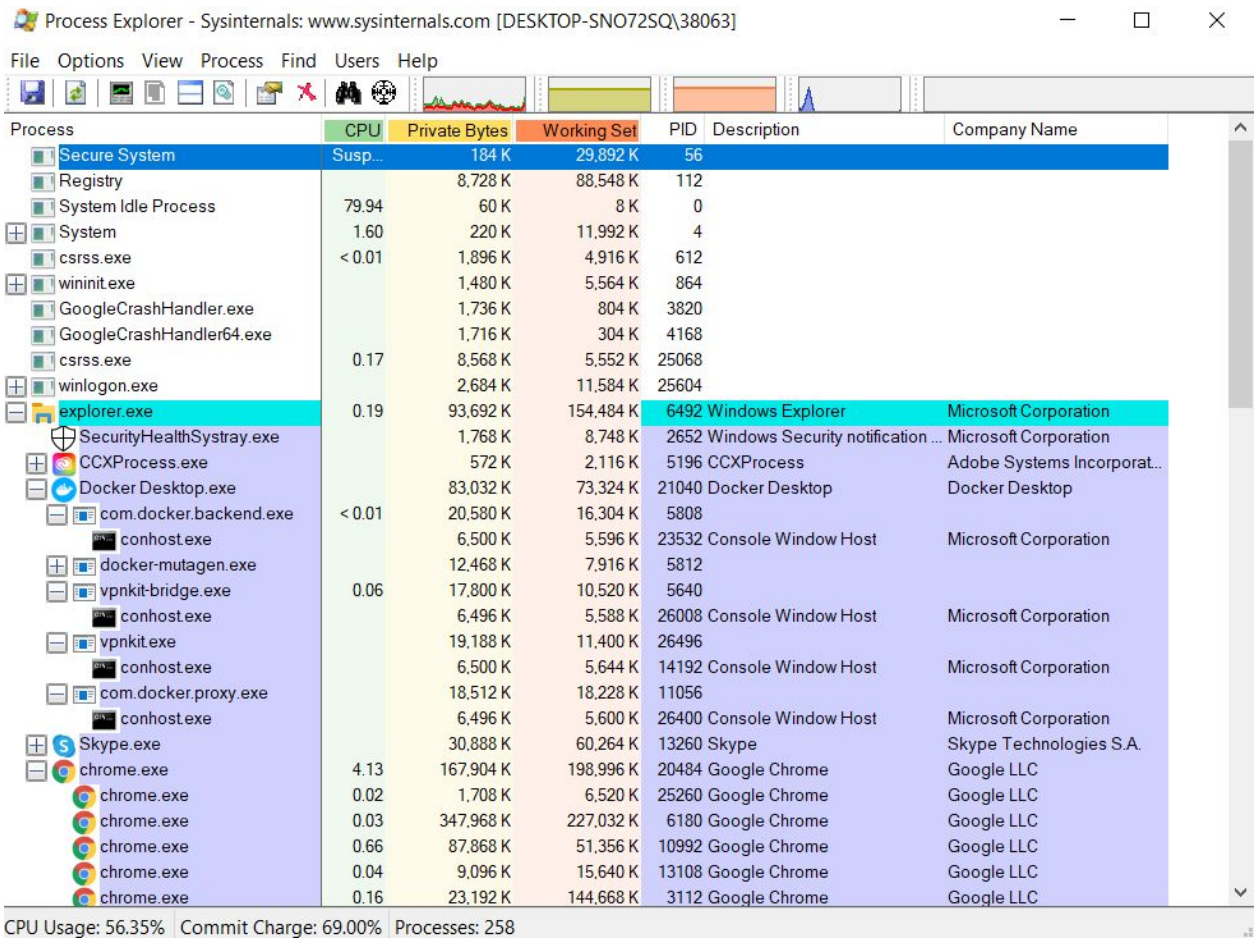


Рис.11 Process Explorer

З допомогою цієї утиліти можна отримати ще більше інформації про процеси і потоки, наприклад:

- Токен безпеки процесу (список груп, привілеї, стан віртуалізації)
- Перелік сервісів у процесах сервіс-хостингу
- Перелік додаткових атрибутів процесу (mitigation policies та рівень їх захисту)
- Процеси, що є частиною job і їх деталі
- Список DLL бібліотек
- Повний список відображених у пам'ять файли (не лише DLL)
- Детальна інформацію про потоки (рис. 12)
- Стек потоку режиму користувача та режиму ядра та інше.

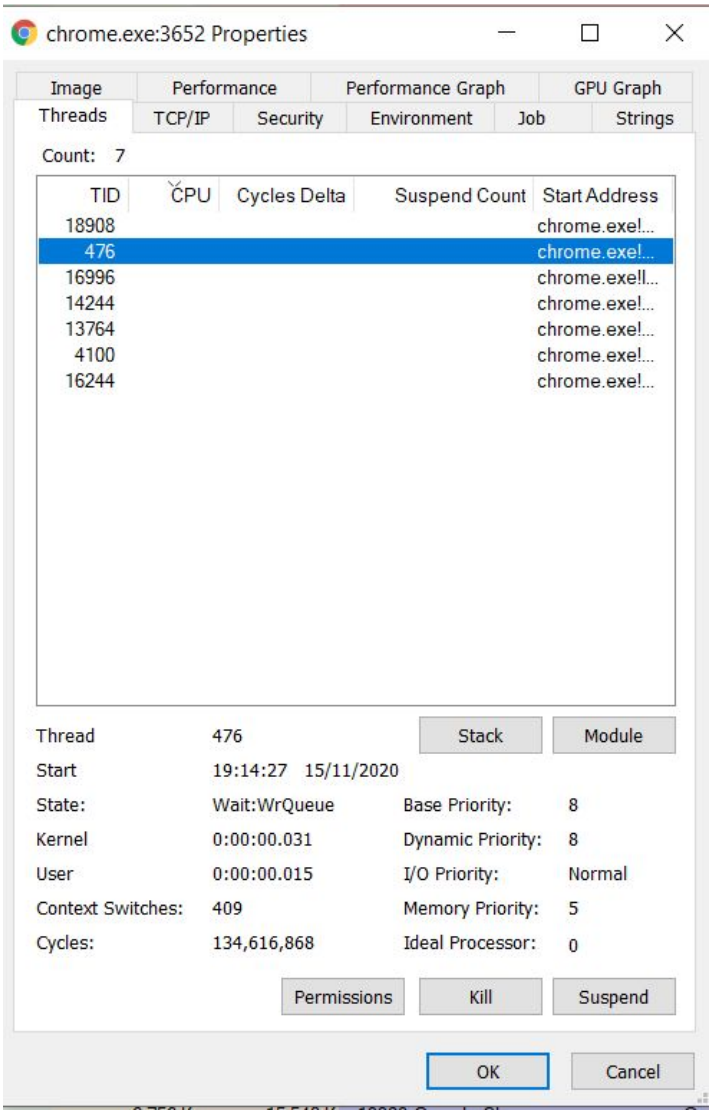


Рис. 12. Інформація про потоки

Process Explorer також надає більше функціональних можливостей:

- Підсвічування, щоб відобразити зміни у процесі, потоці, бібліотеках і списку дескрипторів
- Можливість призупинити процес або потік (рис.13)
- Можливість вбити окремий потік
- Відкрити дескриптори в процесі, включаючи неіменновані

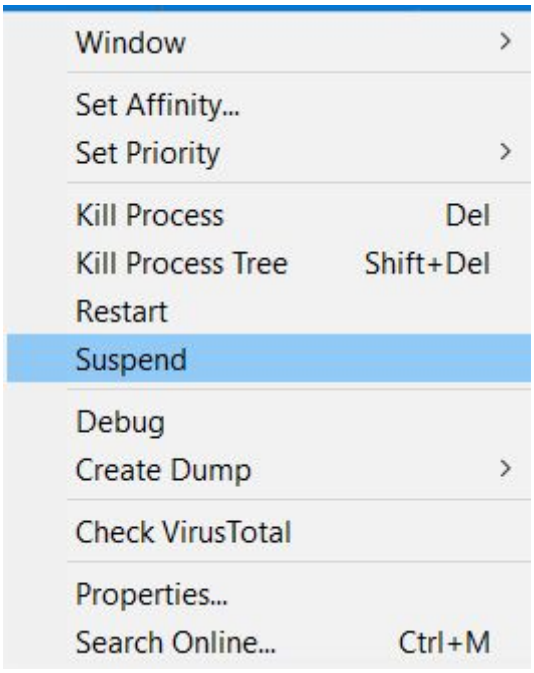


Рис. 13. Призупинити процес

Працюючи з процесами і потоками на різних ОС, буває корисно перевірити наскільки швидше працює програма на одному чи декількох процесорах (ядрах), а також наскільки ефективно вона “розпаралелена”.
Для того, щоб визначити потенційне прискорення алгоритму/програми при збільшенні числа процесорів існує закон Амдала, який виражається наступною формулою:

$$S = \frac{1}{p+\frac{1-p}{n}}$$

де S − прискорення програми (як відношення до її початкового часу роботи); p− частина програмного коду яку можна виконувати послідовно; 1-p – частина програмного коду, яка виконується паралельно; n − кількість процесорів. Якщо послідовна частина програми виконується 10 % всього часу роботи, неможливо прискорити виконання такої програми більше ніж в 10 разів — незалежно від того, скільки процесорів використовує програма.

Щоб визначити реальне прискорення програми потрібно визначити значення A із відношення

$$A = \frac{T_1}{T_n}$$

де T_1 − це час виконання програми в мілісекундах, на одному процесорі (ядрі), а T_n − це час виконання програми в мілісекундах, на n процесорах (ядрах). І знаючи прискорення A ми можемо знайти 1-p – частина програмного коду, яка виконується паралельно, просто прирівнявши $A = S$ і виразивши p.

Задати кількість ядер на яких буде виконуватись наша програма можна за допомогою Set Affinity у Task Manger для Windows

ЗАВДАННЯ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ.

1. За допомогою утиліти «Диспетчер задач» та Process Explorer отримати повну інформацію про процеси: ідентифікатор процесу, завантаження ЦП (центрального процесора), час ЦП, базовий пріоритет, стан процесу, пам'ять-використання, пам'ять-зміни, пам'ять-максимум, помилок сторінки, об’єкти USER, код сеансу, об’єм віртуальної пам’яті, лічильник дескрипторів, лічильник потоків.

2. За допомогою утиліти Process Explorer отримати додаткову інформацію про процеси та їхні потоки.

3. Використовуючи «Диспетчер задач» та Process Explorer змінити пріоритет будь-якого процесу, від низького до «реального часу»; задати відповідність виконання процесів на окремих ядрах центрального процесора; виконати завершення процесу.

4. Використовуючи Process Explorer призупинити процес і відновити його роботу.

5. Скомпілювати файл main.cpp представлений нижче і запустити виконуваний файл на різній кількості активних процесорів (ядер). Знайти для даної програми величини A, S, p при різних вхідних значеннях величини n.

```
//main.cpp
#include <iostream>
#include <thread>
#include <mutex>
#include <vector>
#include <algorithm>
```

```
#include <cmath>
#define DEF_THREAD_COUNT 4
const int N = 50000000;
namespace MCConcurrency
{
   inline int threadsCount(int iterationsCount = -1)
   {
      int numThreads = static_cast<int>(std::thread::hardware_concurrency());
      if (numThreads == 0)
         numThreads = DEF_THREAD_COUNT;
      if (iterationsCount == 0)
         return 1;
      if (iterationsCount > -1)
         numThreads = std::min(static_cast<int>(numThreads), iterationsCount);
      return numThreads;
   }
   template<typename Function>
   void parallel_for(int first, int last, int threadsNumbers, const Function& l)
   {
      int numThreads = threadsNumbers;
      int elementsCount = last - first;
      //auto numThreads = threadsCount(last - first);
      if (numThreads <= 0)
         return;
      int threadNumber = 0;
      std::mutex tasksMutex;
      std::vector<std::thread> threads;
      const auto chunk = std::max(static_cast<int>(std::ceil(elementsCount /
static_cast<double>(numThreads))), 1);
      for (int i = 0; i < numThreads; i++)
      {
         threads.push_back(std::thread([&]()
         {
            int thread_number;
            tasksMutex.lock();
            thread_number = threadNumber;
            ++threadNumber;
            tasksMutex.unlock();
            int beg = first + thread_number * chunk;
            int end = std::min(first + (thread_number + 1)*chunk, elementsCount);
            for (int ind = beg; ind < end; ind++)
            {
               l(ind, thread_number);
            }
         }));
      }
      std::for_each(begin(threads), end(threads), [](std::thread& th)
      {
```

```
            th.join();
        });
    }
}
int main()
{
    typedef std::chrono::high_resolution_clock Time;
    typedef std::chrono::milliseconds ms;
    typedef std::chrono::duration<float> fsec;
    std::vector<float> arr(N);
    auto numThreads = MCConcurrency::threadsCount(N);
    std::cout << "Set process affinity (cores count) and press <Enter>";
    getchar();
    auto t0 = Time::now();
    MCConcurrency::parallel_for(0, N, numThreads, [&](int nIndex, int threadNumber) {
        arr[nIndex] = sin(threadNumber) * cos(threadNumber);
    });
    auto t1 = Time::now();
    fsec fs = t1 - t0;
    ms d = std::chrono::duration_cast<ms>(fs);
    std::cout << "Duration: " << d.count() << "ms\n";
    return 0;
}
```

6. Дослідити вплив зміни відповідності ядру на швидкодію процесу. Виконати завдання згідно варіанту, що відповідає порядковому номеру у списку підгрупи.

Варіанти завдань:

1) Пошук файлів
2) Стискання файлів
3) Компілювання проекту з використанням IDE
4) Завантаження веб сторінок
5) Завантаження файлів за допомогою http протоколу
6) Передавання файлу мессенджером
7) Сканування деякої папки антивірусом
8) Видалення файлів
9) Копіювання файлів
10) Скачування за допомогою BitTorrent протоколу
11) Рендеринг в будь-якому графічному редакторі
12) Відмальовування сцени (fps) деякої відеогри (неонлайн)
13) Відмальовування сцени (fps) деякої відеогри (онлайн)
14) Робота сервера баз даних або веб сервера
15) Роздача за допомогою BitTorrent протоколу

7. Результати лабораторної роботи оформити у звіт, у висновку надати порівняння моніторингу процесів у різних системах різними утилітами.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Операційна система це одна програма чи сукупність програм?
2. Які функції операційної системи?
3. Що таке обчислювальний процес?
4. Чому “програма” “не дорівнює” “процес”?
5. Що таке блок управління процесом (PCB)?
6. Який атрибут ідентифікує процес у ОС Windows?
7. Яка структура процесу в ОС Windows?
8. Як представили дерево процесів ОС Windows?
9. Як формується пріоритет в ОС Windows?
10. Яка особливість встановлення пріоритету потоку в ОС Windows?
11. Загальна модель станів процесу?
12. За допомогою яких засобів можна моніторити процеси в операційних системах?
13. Назвіть вкладки в Диспетчері задач
14. У якій вкладці Диспетчера задач можна отримати детальну інформацію про процеси?
15. Які значення можна побачити у стовпці Status?
16. Чи завжди стан Running буде означати що процес справді активний і не перебуває в очікуванні?
17. Інформацію про яку пам’ять можна отримати з Диспетчера задач?
18. Що таке Дескриптори?
19. Що входить до Дескрипторів?
20. Чи можна в Диспетчері задач дізнатись про потоки і взаємодіяти з ними?
21. Чим відрізняються переміщувані і непереміщувані пули?
22. Яку додаткову інформацію можна отримати з утиліти Process Explorer?
23. Що відображають кольори в Process Explorer?
24. Як можна швидко найти необхідний нам процес?
25. В якій утиліті є можливість керування потоками?

26. Як дізнатися пріоритет потоку?
27. Що виражає закон Амдала?

СПИСОК ЛІТЕРАТУРИ

1. Демида Б.А., Обельовська К.М., Яковина В.С. Основи адміністрування LAN у середовищі MS Windows : навч. посіб. — Львів: Видавництво Львівської політехніки, 2013. — 488 с.
2. Дендюк М.В., Рожак П.І., Семенишин Н.О. Методика збільшення швидкості паралельного рендерингу за допомогою bittorrent протоколу. Збірник науково-технічних праць. Науковий вісник НТЛУ, 2018, т.28, №8, с. 132-135
3. Основи системного адміністрування. Електронний навчально-методичний комплекс Е41-163-03/2012 від 03.04.2012 / Укл.: Яковина В.С.
4. Робачевский А. М. Операционная система UNIX. СПб.:БХВ-Петербург, 2002. 528 с.
5. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015, 1120 с.
6. Silberschatz A., Galvin P. B., Gagne G. Operating system concepts: 10th edition. Hoboken, NJ : Wiley, 2018, 1278 P.
7. Yosifovich P., Ionescu A., Russinovich M.E., Solomon D.A. Windows Internals: Seventh Edition Part 1. System architecture, processes, threads, memory management, and more. Washington: Microsoft Press, 2017, 1120 P.

НАВЧАЛЬНЕ ВИДАННЯ

ОЗНАЙОМЛЕННЯ ТА КЕРУВАННЯ ПРОЦЕСАМИ В ОПЕРАЦІЙНИХ СИСТЕМАХ ДЛЯ ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА. WINDOWS.

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи №1
з дисципліни «Операційні системи»
для студентів першого (бакалаврського) рівня вищої освіти
спеціальності 121 «Інженерія програмного забезпечення»

Укладачі: Яковина Віталій Степанович, д-р. техн. наук, проф.
Грицай Оксана Дмитрівна, канд. фіз.-мат. наук,
Майхер Вікторія Юріївна, канд. техн. наук
Семенишин Назар Олегович