МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Д. Ю. Голубничий
А. В. Холодкова

ОПЕРАЦІЙНІ СИСТЕМИ

Навчальний посібник

Харків
ХНЕУ ім. С. Кузнеця
2018

УДК 004.451(075)
Г62

Авторський колектив: канд. техн. наук, доцент Д. Ю. Голубничий – вступ, розділи 1, 3, 4; канд. техн. наук, доцент А. В. Холодкова – розділ 2.

Рецензенти: заступник начальника кафедри математичного та програмного забезпечення АСУ Харківського національного університету Повітряних Сил імені Івана Кожедуба, д-р техн. наук, старший науковий співробітник *С. А. Олізаренко*; доцент кафедри ПІІТУ Національного технічного університету "Харківський політехнічний інститут", канд. техн. наук, доцент *О. В. Шматко*.

Рекомендовано до видання рішенням ученої ради Харківського національного економічного університету імені Семена Кузнеця.
Протокол № 5 від 29.01.2018 р.

Самостійне електронне текстове мережеве видання

Голубничий Д. Ю.

Г62 Операційні системи [Електронний ресурс] : навчальний посібник / Д. Ю. Голубничий, А. В. Холодкова. – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 317 с.

ISBN 978-966-676-761-8

Подано лабораторні роботи із навчальної дисципліни з метою закріплення теоретичних знань і набуття практичних навичок у роботі з обчислювальною технікою.

Рекомендовано для студентів, викладачів і користувачів, які вивчають основи побудови операційних систем та їх практичне застосування під час створення додатків для різних галузей економіки.

УДК 004.451(075)

ISBN 978-966-676-761-8

Вступ

Навчальний посібник призначено для проведення лабораторних робіт із навчальної дисципліни "Операційні системи". Лабораторне заняття – це форма навчального заняття, за якої студенти під керівництвом викладача особисто проводять натурні або імітаційні експерименти чи досліди, із метою практичного підтвердження окремих теоретичних положень навчальної дисципліни, набувають практичних навичок у роботі з обчислювальною технікою, оволодівають методикою експериментальних досліджень у конкретній предметній області. Лабораторні заняття з навчальної дисципліни "Операційні системи" проводять у спеціально оснащеному навчальному класі з використанням комп'ютерного обладнання, пристосованого до навчального процесу.

Об'єктом вивчення навчального матеріалу з навчальної дисципліни є типові механізми, технології, протоколи взаємодії різноманітних компонент операційної системи як на рівні ядра, так і на рівні користувача.

Предметом вивчення навчального матеріалу є теоретичні концепції та методології, принципи функціонування, вибору та практичної реалізації складових частин операційної системи.

Із метою підвищення якості навчального процесу, під час проведення лабораторного заняття навчальну групу розподіляють на дві підгрупи. Кожний студент працює самостійно, виконуючи загальні та індивідуальні завдання для лабораторного дослідження.

Практичний модульний контроль здійснюють після виконаних лабораторних завдань у межах відповідного модуля з урахуванням захищених звітів із лабораторних робіт.

Необхідним елементом успішного засвоєння навчального матеріалу цієї навчальної дисципліни є самостійна робота студентів із науково-технічною літературою та сучасними технічними засобами комп'ютеризованого оброблення інформації.

Розділ 1. Архітектура операційних систем

Лабораторна робота 1
Дослідження операційної системи ReactOS

Мета роботи: ознайомлення з основними принципами встановлення (інсталяції) операційної системи ReactOS і її конфігурації на віртуальних машинних середовищах, а також набуття практичних навичок у побудові й дослідженні ReactOS і процесі інсталяції.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно вивчити принципи й логіку роботи ReactOS-інсталятора, його основні параметри під час запуску в режимі командного рядка. Навчитися виконувати підготовчі операції на початковому етапі встановлення ReactOS. Особливу увагу слід приділити процесу конфігурації операційної системи (ОС) під час установлення і після завершення кінцевого етапу. Вивчити вимоги до апаратного забезпечення ОС.

Додаткову інформацію під час підготовки до роботи можна отримати в [24; 25; 28].

Теоретичні відомості

ReactOS – це спроба розробити клон Windows із відкритим початковим кодом. Як зразок для копіювання було вибрано Microsoft Windows NT 4.0. Перед розробниками стояла мета не просто створити середовище, у якому б запускалися Windows-додатки, але й написати повноцінну операційну систему, сумісну з Windows NT на рівні як додатків, так і драйверів. Незважаючи на те що як зразок була вибрана NT 4.0, розробники завжди озираються на пізніші версії Windows: 2000 і XP. Архітектура NT дозволяє використовувати підсистеми, так само як і архітектура ReactOS. **Підсистеми** – це реалізація application programming interface (API) інших ОС, що дозволяє запускати в ReactOS їхні додатки. Зараз до розроблення готують підсистеми Java, OS/2,DOS, POSIX, у майбутньому, можливо, інші. ReactOS завжди планували розвивати за участю проекту WINE,

щоб, за потреби, використовувати накопичений ними досвід і не переписувати одне й те саме. Це в основному стосується призначеної для користувача частини ОС, та одного разу ядро ReactOS буде повнішим, ніж ця його частина.

На сьогодні ReactOS здатний завантажувати ReactOS Explorer (аналог провідника Windows), запускати Win32-додатки, завантажувати та використовувати драйвери від Windows NT, 2000, XP, 2003; працювати на x86 платформі й підтримувати ігрову консоль Xbox.

ReactOS підтримує файлові системи FAT12/16/32 і ISO-9660 (CD-ROM).

Список програм ReactOS, що працюють під системою:

1) офісні (AbiWord, MS Office 7, MS Works 4, Foxit Reader);

2) Інтернет (Firefox, SeaMonkey, Thunderbird, Adobe Flash Player, mIRC, PUTTY, ULTRAVNC);

3) графіка (Adobe Photoshop 3 – 5, Google Picasa, Paint Shop 4).

Із проблем можна зазначити відсутність стабільної підтримки USB-пристроїв, непрацездатність деяких драйверів пристроїв, програм, загальну нестабільність системи (витоки пам'яті), повільну роботу деяких частин (зокрема, підсистеми Win32).

Увесь процес установлення (інсталяції) і конфігурація ReactOS складається з декількох етапів.

Етап 1. Підготовка до інсталяції

ОС ReactOS має достатньо не високі вимоги до апаратної конфігурації комп'ютера, порівняно з ОС клону Windows. Її можна тільки встановити на апаратній платформі x86.

Мінімальні вимоги до x86 комп'ютера для того, щоб на ньому можна було встановити ReactOS, такі:

1. Мінімальна частота процесора – 600 MHz;
2. Рекомендована частота процесора – 1.6 GHz;
3. Мінімальний обсяг ОЗУ – 128 MB;
4. Рекомендований обсяг ОЗУ – 256 MB;
5. Вільний дисковий простір – 2 ГБ;
6. Вимога до монітора:

діагональ – не нижча за 17';

мінімальна роздільна здатність монітора – 1024 × 768;

глибина палітри – 65 535 кольорів.

7. Наявність маніпулятора – "мишки".
8. Клавіатура стандартна.

Етап 2. Створення нової віртуальної машини у VMWare

Дистрибутив операційної системи ReactOS має розмір близько 50 Мбайт, оскільки в дистрибутиві містяться тільки найнеобхідніші компоненти – ядро, бібліотеки і декілька стандартних додатків. Під час створення нової віртуальної машини в розділі вказівки типу ОС, яку буде інстальовано, необхідно вказати Windows NT, тому що ReactOS – клон Windows NT 4.0.

Після створення віртуальної машини з CD-ROM необхідно встановити образ *.iso з дистрибутивом ReactOS (файл ReactOS*.iso). Після запуску віртуальної машини завантажиться iso-образ, починається встановлення системи. На рис. 1.1 – 1.7 наведено скриншоти встановлення системи ReactOS на емуляторі VMware.

Етап 3. Первинне налаштування встановлення (інсталяції) системи ReactOS

Після під'єднання ReactOS.iso і запуску віртуальної машини з'являється така початкова текстова сторінка встановлення:

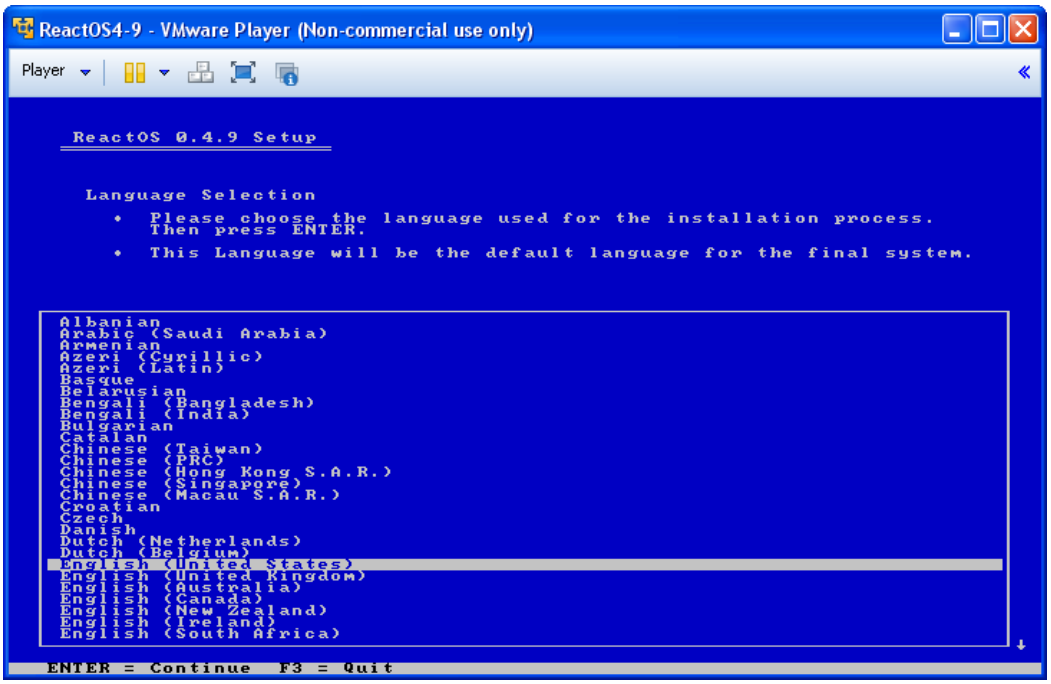


Рис. 1.1. **Вибір мови системи за замовчуванням**

На цьому етапі встановлюють налаштування монітора, клавіатури (український розклад відсутній), тип комп'ютера.

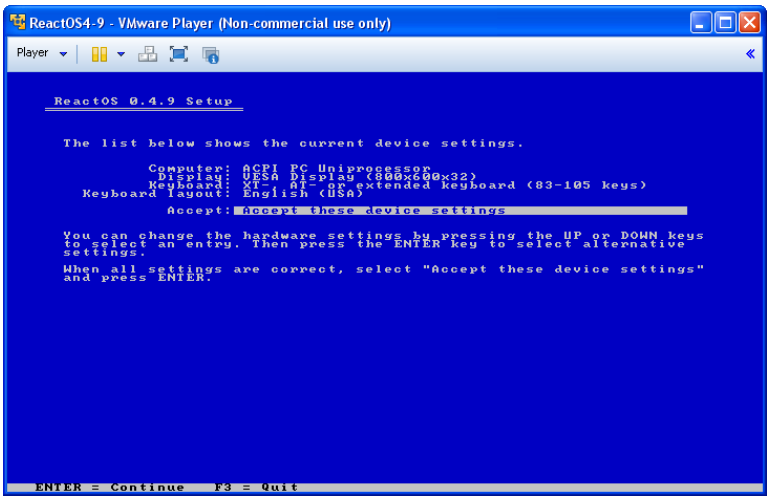


Рис. 1.2. **Вікно задавання налаштування пристроїв**

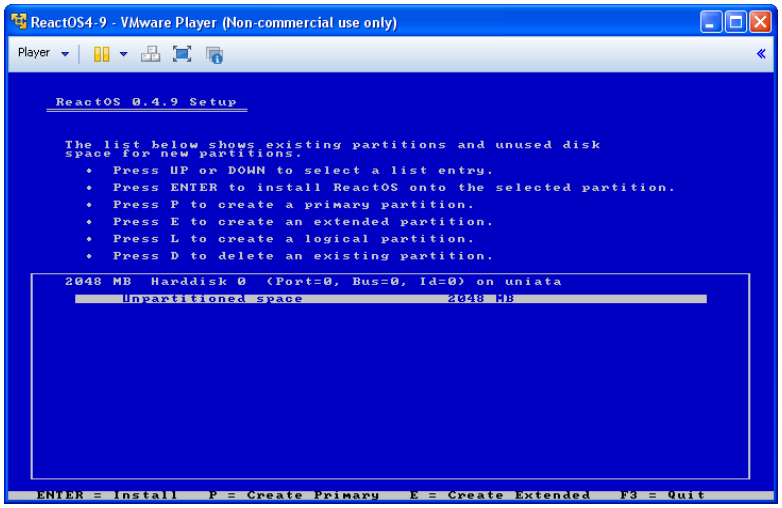


Рис. 1.3. **Вікно вибору диска, на який буде зроблено встановлення**

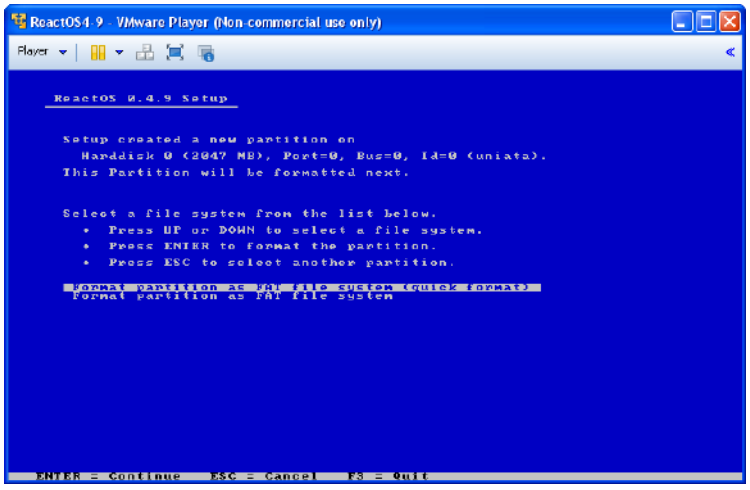


Рис. 1.4. **Вікно вибору типу файлової системи**

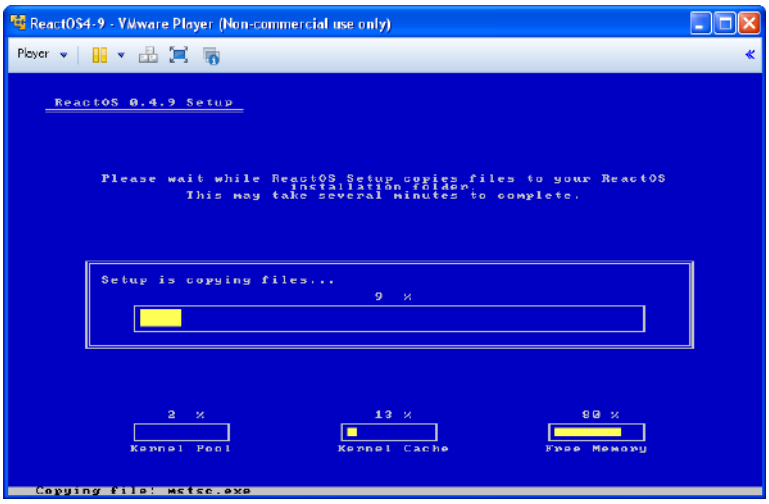


Рис. 1.5. **Вікно індикації копіювання файлів інсталятора**

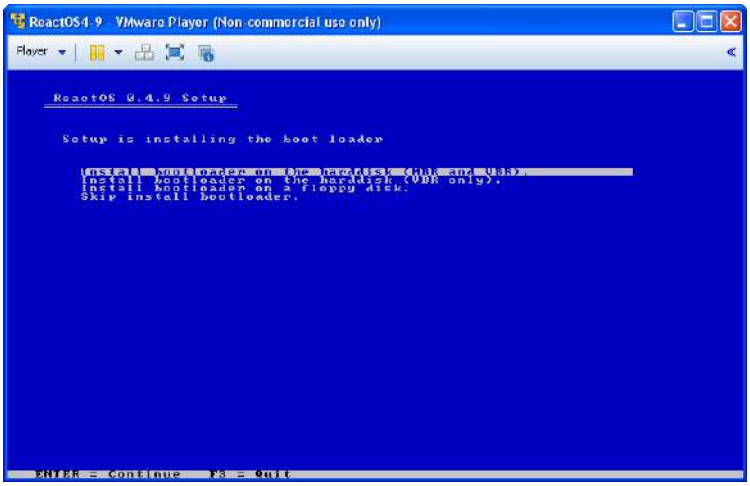


Рис. 1.6. **Вікно вибору місця встановлення завантажувального сектора**

Рис. 1.7. **Завершення первинного налаштування встановлення та автоматичне перезавантаження системи**

Етап 4. Вторинне налаштування встановлення (інсталяції) системи ReactOS

Після завершення первинного встановлення відбувається перезавантаження системи в автоматичному режимі (рис. 1.8).

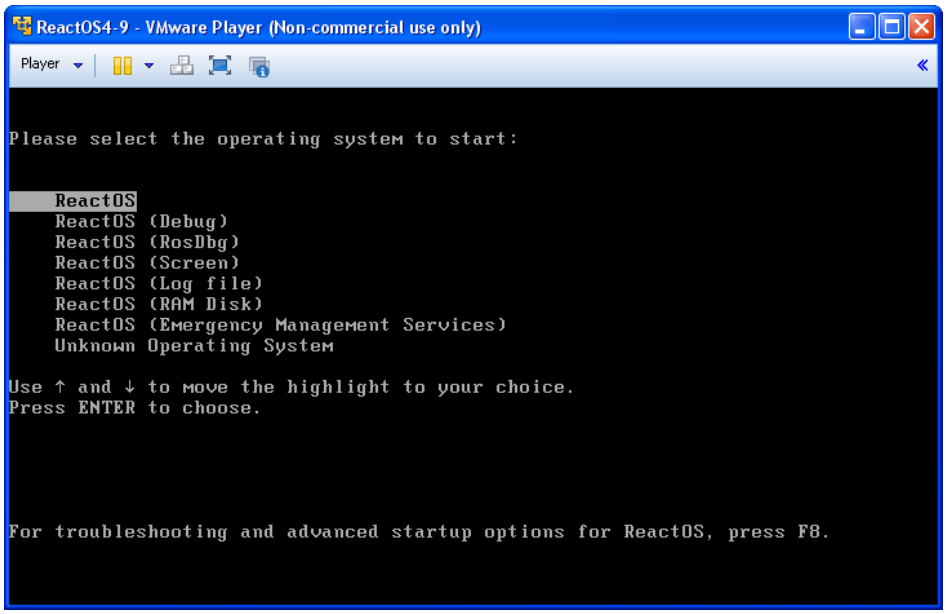


Рис. 1.8. **Вибір варіанта завантаження системи**

Після запуску системи ReactOS виявляє, що працює всередині віртуальної машини. Далі надає запит на встановлення драйвера монітора (рис.1.9).

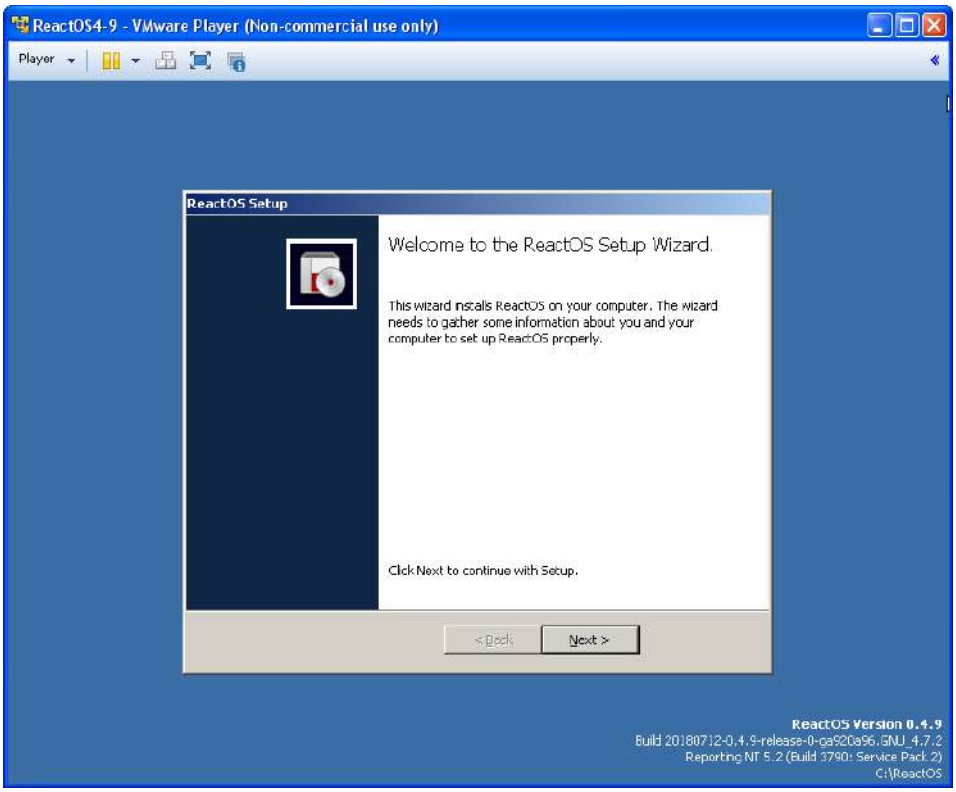


Рис. 1.9. ***Майстер установлення*** **ReactOS для персонального налаштування**

Із метою безпечного виконання встановлення незнайомої та недопрацьованої системи запит було відхилено. І нарешті, після вказівки імені користувача-адміністратора, його пароля, імені комп'ютера й організації, а також вибору налаштування мов, дати, часу й часового поясу здійснюють первинне завантаження *Робочого столу* операційної системи ReactOS (рис.1.10).

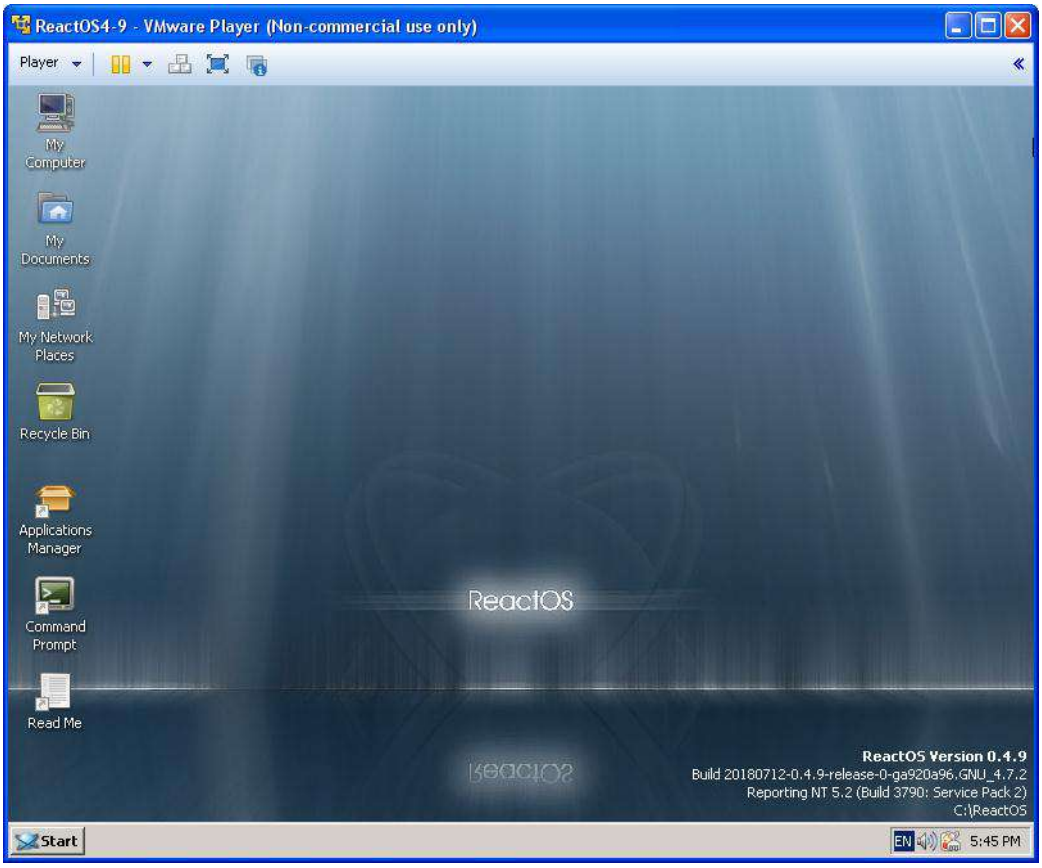


Рис. 1.10. ***Робочий стіл*** **системи ReactOS**

Інтерфейс системи нагадує інтерфейс Windows. Операційна система містить набір основних службових утиліт. Їх менше, ніж у дистрибутиві Windows. Наявні лише основні інструменти, без яких не обійтися:

провідник (explorer.exe);
редактор реєстру (regedit.exe);
диспетчер завдань (taskmgr.exe);
блокнот (notepad.exe);
графічний редактор (mspaint.exe);
калькулятор (calc.exe);
оболонка командного рядка (cmd.exe).

Також наявні ігри – аналоги "Сапера", "Павука" і "Пасьянсу" у Windows. Багато елементів інтерфейсу не допрацьовано: є тільки форма, а функції, що мають їм відповідати, ще не реалізовано.

На цьому встановлення завершується.

Загальні завдання для виконання

1. Розгляньте різні варіанти завантаження системи на вже встановленій віртуальній машині:

ReactOS;

ReactOS (Debug);

ReactOS (RosDbg);

ReactOS (Screen);

ReactOS (Log file);

ReactOS (RAM Disk);

ReactOS (Emergency Management Service).

2. Дослідіть різні варіанти розширеного завантаження ReactOS під час натиснення клавіші **F8**:

Safe Mode – безпечний режим роботи ОС, що обмежує доступ користувача до деяких можливостей системи, для забезпечення стабільності роботи;

Safe Mode with Networking – це безпечний режим, що надає доступ до мережевих можливостей комп'ютера;

Safe Mode with Command Prompt – це безпечний режим, що надає доступ до командного рядка;

Enable Boot Logging – це режим, що дозволяє здійснити протоколювання завантаження операційної системи;

Enable VGA Mode – це режим, що дозволяє перемикатися на інший графічний режим роботи;

Last Known Good Configuration – це режим, що дозволяє відновити систему до стану за останнього працездатного завантаження;

Directory Services Restore Mode (DSRM) – це режим, призначений для відновлення Active Directory. Наприклад, якщо Active Directory пошкоджений і потребує виправлення. Часто DSRM використовують для скидання забутих паролів користувачів та адміністратора домена;

Debugging Mode – це режим, що дозволяє запустити режим налаштування операційної системи;

Custom Boot – це режим, що дозволяє вибрати інші способи завантаження системи, наприклад, завантаження вибраного диска, розділу системи, із файла завантажувального сектора, вибраних компонентів ReactOS, завантаження ОС Linux, якщо відомий шлях до неї.

3. Визначте призначення, параметрів виклику і прикладу використання вбудованих типових команд інтерпретатора командного рядка, відповідно до номера індивідуального варіанта (табл. 1.1).

Таблиця 1.1

Індивідуальні завдання для виконання

Варіанти	Команди інтерпретатора		Розділи	Обчислення
1	2		3	4
1	CMD	POPD	Games & Fun	Сума позитивних елементів масиву
2	RMDIR	HISTORY	Finance	Сума елементів масиву, розташованих між першим та останнім негативними елементами
3	RENAME	CLS	Engineering	Добуток елементів масиву, розташованих між максимальним і мінімальним елементами
4	GOTO	VERIFY	Internet & Network	Сума елементів масиву, розташованих між першим та останнім позитивними елементами
5	CHCP	START	Office	Сума елементів масиву з непарними номерами
6	ECHO	MKLINK	Graphics	Максимальний елемент масиву
7	ALIAS	LABEL	Development	Номер максимального елемента масиву
8	CALL	IF	Office	Сума модулів елементів масиву, розташованих після мінімального за модулем елемента
9	SET	CHOICE	Engineering	Сума елементів масиву, розташованих після першого позитивного елемента
10	PUSHD	SCREEN	Graphics	Суму негативних елементів масиву

Продовження табл. 1.1

1	2		3	4
11	BEEP	FOR	Games & Fun	Сума елементів масиву, розташованих між першим і другим негативними елементами
12	MOVE	CD	Internet & Network	Номер мінімального за модулем елемента масиву
13	DATE	ATTRIB	Development	Мінімальний елемент масиву
14	TIME	ERASE	Internet & Network	Добуток елементів масиву, розташованих між максимальним за модулем і мінімальним за модулем елементами
15	FREE	VOL	Edutainment	Кількість елементів масиву, більших ніж число K
16	DELETE	TYPE	Graphics	Максимальний за модулем елемент масиву
17	PROMPT	REPLACE	Office	Добуток елементів масиву, розташованих після максимального за модулем елемента
18	COLOR	VER	Games & Fun	Сума модулів елементів масиву, розташованих після першого негативного елемента
19	MKDIR	BEEP	Development	Номер мінімального елемента масиву
20	DIR	DELETE	Edutainment	Кількість негативних елементів масиву
21	REN	MOVE	Internet & Network	Добуток елементів масиву, розташованих між першим і другим нульовими елементами
22	PATH	DATE	Office	Сума цілих частин елементів масиву, розташованих після останнього негативного елемента
23	ATTRIB	CD	Games & Fun	Сума позитивних елементів масиву, розташованих до максимального елемента
24	DIR	TYPE	Engineering	Кількість елементів масиву, менших ніж число K
25	MKDIR	FOR	Graphics	Сума елементів масиву, розташованих до мінімального елемента
26	VOL	PROMPT	Games & Fun	Сума елементів масиву, розташованих до останнього позитивного елемента

1	2		3	4
27	REPLACE	TIME	Internet & Network	Сума елементів масиву, розташованих між першим і другим позитивними елементами
28	RD	PATH	Edutainment	Добуток позитивних елементів масиву
29	COPY	REN	Office	Номер максимального за модулем елемента масиву
30	ERASE	FREE	Development	Добуток негативних елементів масиву

4. Виконайте встановлення одного додатка (на власний вибір студента) у ReactOS Application Manager зі вказаного в табл. 1.1 розділу завдання.

Наприклад. На рис. 1.11 виконано встановлення браузера Mozilla Firefox у розділ Internet & Network.

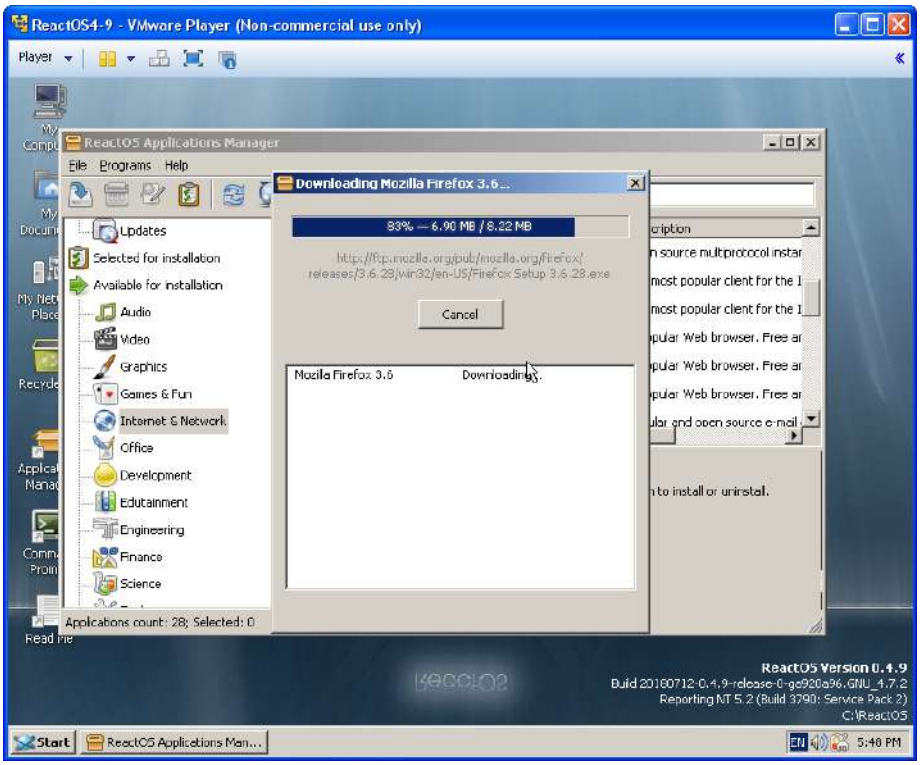


Рис. 1.11. **Процес інсталяції браузера Mozilla Firefox**

Результати встановлення відобразіть у звітах зі вказівкою алгоритму встановлення, його особливостей та описом послідовності дій під час виконання кожного етапу завдань.

Додаткові завдання

1. Виконайте встановлення одного з вільно поширюваних компіляторів мови C++, наприклад, Dev-C++ (рис.1.12).

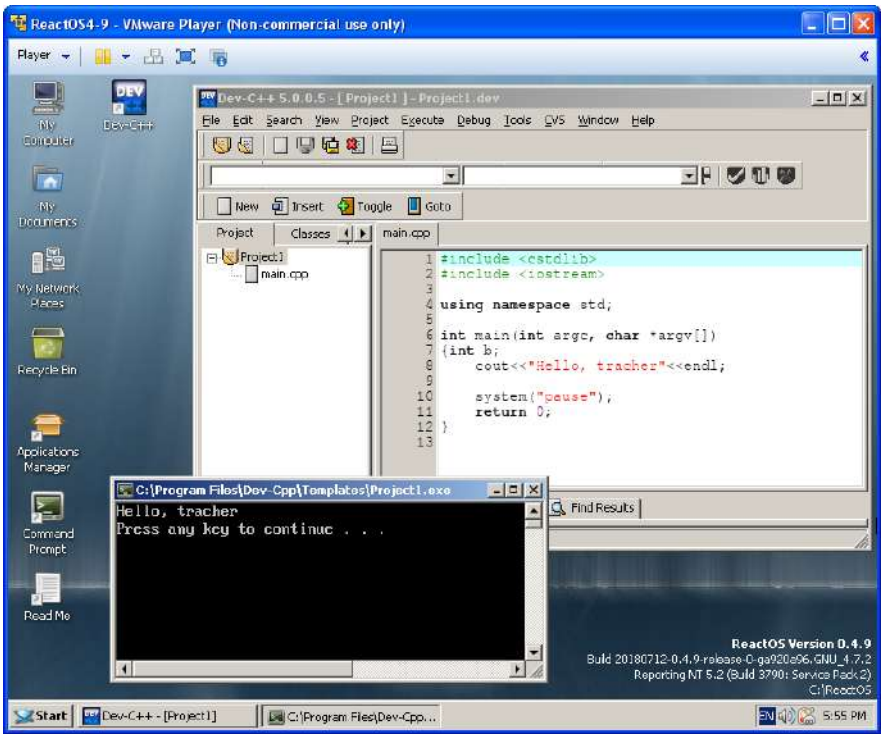


Рис. 1.12. **Приклад створення програми в Dev-C++**

2. Розробіть програму виконання обчислення, заданого в табл. 1.1, в одновимірному масиві, що складається з *n* речових елементів.

Контрольні запитання

1. Визначте мету використання ОС.
2. У чому полягають принципові відмінності ОС ReactOS і версій ОС Windows?
3. Які особливості інсталяції ОС ReactOS?
4. Назвіть основні принципи побудови ОС.
5. У чому відмінність режимів завантаження ОС ReactOS і ОС Windows?
6. Назвіть основні версії ReactOS.
7. У чому полягають специфічні особливості командного рядка ОС ReactOS?
8. Назвіть відмінності створення програм у середовищі Dev-C++.

9. До якого типу проекту належить операційна система ReactOS?

10. Якою є операційна система ReactOS щодо Windows?

11. Який тип ядра в ReactOS?

12. Назвіть переваги ReactOS, порівняно з Windows?

13. Назвіть частину ядра ReactOS, сумісну з Windows?

14. Назвіть компоненту ReactOS, яка виконується у призначеному для користувача режимі?

15. Що покладено в основу виконавчої підсистеми ReactOS?

16. Що в ReactOS належить до завантажуваних модулів ядра?

17. Яку мову програмування використовують для написання ядра ReactOS?

18. Що забезпечує рівень абстрагування від обладнання в ReactOS для апаратного забезпечення?

19. Через який механізм *Диспетчер* PnP у ReactOS зв'язується із драйверами?

20. У якому режимі працює завантажувач ReactOS?

21. Який елемент архітектури операційної системи ReactOS відповідає за системні ресурси?

22. Який рівень ядра становить у ReactOS виконавча підсистема?

23. Який елемент виконавчої системи ReactOS відстежує об'єкти процесів і потоків?

24. Що убезпечує процеси ReactOS?

Лабораторна робота 2
Дослідження операційної системи KolibriOS

Мета роботи: ознайомлення з основними принципами встановлення (інсталяції) операційної системи KolibriOS і її конфігурації на віртуальних машинних середовищах, а також набуття практичних навичок у побудові та дослідженні KolibriOS у процесі інсталяції.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно вивчити принципи й логіку роботи інсталятора KolibriOS, його основні параметри під час запуску в режимі командного рядка. Навчитися виконувати підготовчі операції на початковому етапі встановлення KolibriOS. Особливу увагу слід приділити процесу конфігурації ОС

під час установлення і після завершення кінцевого етапу. Вивчити вимоги до апаратного забезпечення ОС.

Додаткову інформацію під час підготовки до роботи можна отримати у [26].

Теоретичні відомості

KolibriOS – це операційна система для комп'ютера, яка повністю написана на асемблері fasm та поширювана на умовах General Public License (GPL). Її створено на основі MenuetOS. Це альтернативна операційна система, оскільки вона використовує власні стандарти й не заснована на POSIX і UNIX. Систему розраховано на використання асемблера для написання додатків, але є і програми, написані на мовах Ада, Сі, C++, Free Pascal, Forth. На сьогодні переважна більшість розробників живуть на території країн колишнього СРСР. Це повноцінна операційна система із графічним інтерфейсом користувача з роздільною здатністю до 1 280×1 024 і 16 мільйонів кольорів, що містять понад 150 додатків (рис. 2.1).

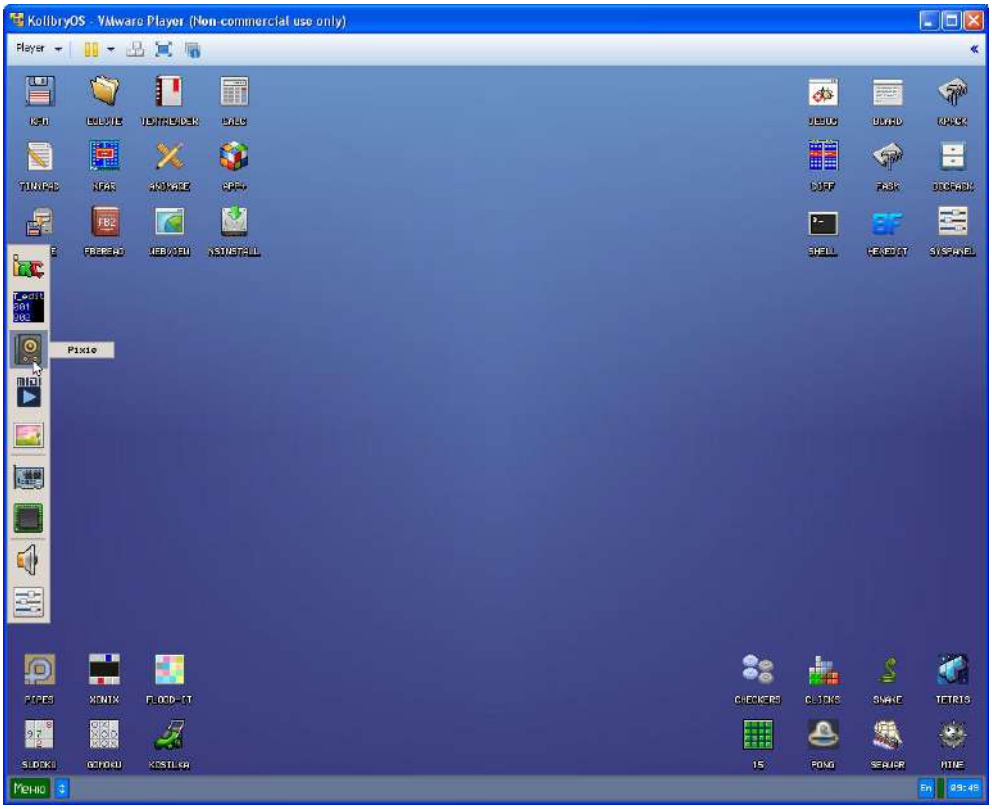


Рис. 2.1. ***Робочий стіл*** **операційної системи KolibriOS**

KolibriOS можна вантажити з дискети, CD-диска і вінчестера. Для завантаження з вінчестера без використання дискети та зміни MBR (Master Boot Record – головного завантажувального запису) доводиться вдаватися до різних хитрощів, залежно від операційної системи. У Windows 95 і 98 можна використовувати `autoexec.bat` і `config.sys` для запуску `meosload.com` – завантажувача KolibriOS і MenuetOS із DOS, а в NT 5.x (і вище) потрібно використовувати файл `boot.ini` для вказівки завантажувального сектора. У цьому разі в корені першого розділу має перебувати файл образу ОС, і там же файл ядра `kernel.mnt`. Також є завантажувач `acroboot`, який не набув великого поширення через інертність звичок користувачів.

Ядро повністю написано на асемблері FASM. Цим досягають високу швидкодію і невеликий розмір системи.

Ядро монолітне з можливістю завантаження зовнішніх драйверів (рис. 2.2). На сьогодні драйвери призначено для підтримання звуку для кодеків AC97 (два варіанти: один драйвер для Intel і NVidia, другий – для SiS) і для підтримання апаратного курсора мишки у відеокарт ATI (для NVidia подібний драйвер ще не написано).

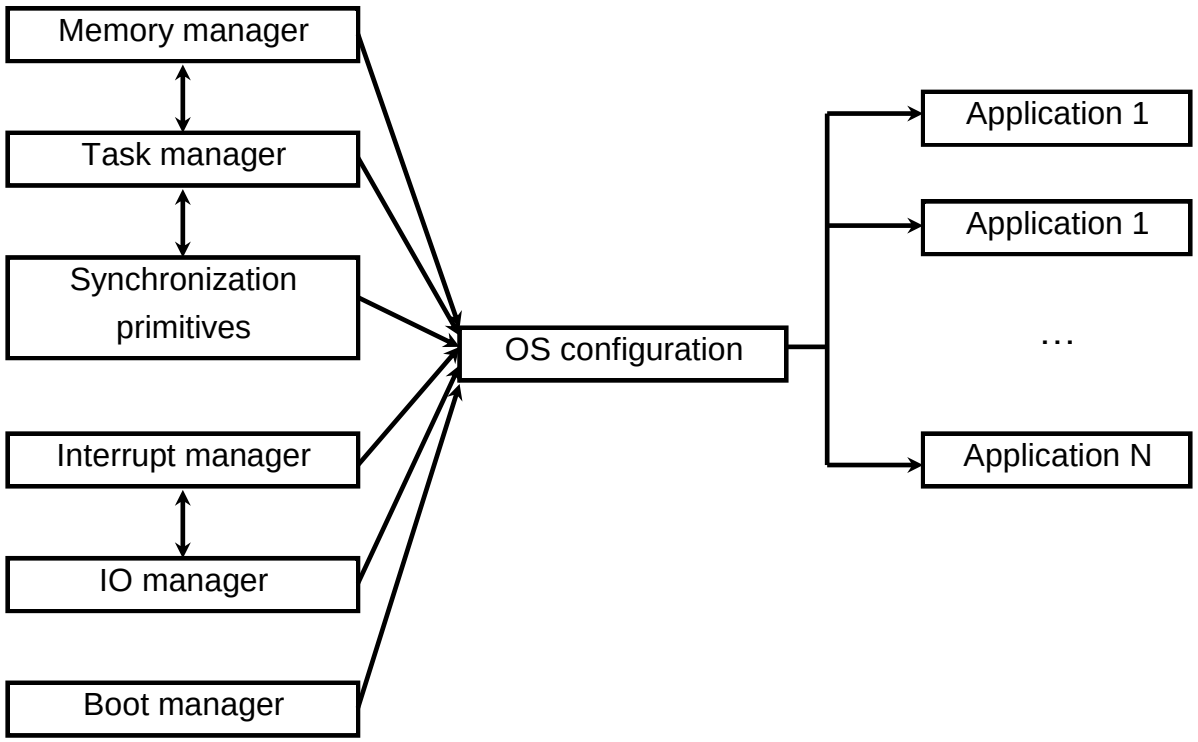


Рис 2.2. **Набір модулів ядра KolibriOS**

Поняття процесу як такого в KolibriOS дуже зачаткове: процес – це об'єднання потоків з одним і тим самим адресним простором. У всіх таких об'єднуваних потоків одне й те саме ім'я та один і той самий розмір використовуваної пам'яті.

Потоки, утім, є і мають такі характеристики (memmap.inc із вихідних кодів ядра):

ідентифікатор (TID), кожному створюваному потоку призначено унікальний ідентифікатор;

стан потоку: активний (виконується прямо зараз або чекає перемикання завдань на нього), заморожений; завершує, той, хто чекає;

вікно: кожен потік має тільки одне вікно, яке може бути невидимим, але обов'язково є;

використання процесора: кількість тактів за останню секунду, яку процесор витратив на виконання саме цього потоку;

ім'я процесу (ім'я виконуваного файла);

маска подій, про які система сповіщає потік;

системний стек;

список об'єктів ядра, що асоціюються із цим потоком;

карта дозволених портів уведення/виведення;

потокова папка для функцій файлової системи;

буфер для повідомлень IPC (наявний, тільки якщо потік його явно визначив).

Процес не ідентифікується, інформацію про потоки всередині ОС зібрано у статичний масив на 255 входів (нумерованих від 0 до 255, причому 0-й слот не можна використовувати, так що всього в системі може бути не більше ніж 255 потоків). Деякі системні функції набирають номера слота, деякі – ідентифікатора.

IPC-взаємодії процесів практично немає – є тільки спеціальна системна функція для передавання даних від процесу-джерела до процесу-приймача, причому приймач має заздалегідь підготувати буфер і чекати цих даних, і деякі можливості з налаштування додатків. Відповідно, синхронізації теж практично немає – один процес може тільки перевірити, чи завершився інший.

Процесорний час розподілено між всіма потоками порівну, кожні 0,01 с ядро передає управління черговому потоку (потік може використовувати свій квант частково, тоді перемикання відбудеться раніше, ніж через указаний інтервал). У цьому сенсі всі потоки рівноправні, включаючи потік операційної системи, який обробляє мишку (рисує курсор і відстежує

події, що відбуваються з мишкою типу переміщення/натискання/натискання на кнопки, визначені додатками), мережу, завершує процеси, переміщає вікна, відстежує натиснення комбінації <Ctrl+Alt+Del>, а у вільний час дає процесору відпочити інструкцією `hlt`.

Для роботи з портами введення/виведення не обов'язково забиратися в ядро і/або писати драйвер. Для цього є спеціальна системна функція, що резервує за додатком запитані ним порти, – звичайно, якщо це не суперечить бажанням самої системи та інших потоків.

Установлення KolibriOS проходить дуже легко і швидко – досить записати образ на дискету та завантажити з неї. Процес запуску KolibriOS полягає в такому: розпакувати архів kolibri.zip на віртуальний жорсткий диск (наприклад, у корінь диска C:\), запустити програму `rawrite2.exe` з розпакованої папки, потім увести `kolibri.img`, написати "`a`" і вставити чисту відформатовану дискету у віртуальний дисковід.

Загальні завдання для виконання

1. Розгляньте процес інсталяції KolibriOS на віртуальну машину.
2. Промоделюйте різні варіанти завантаження системи. Продемонструйте скриншотами їхні особливості:

на підставі **iso-образу** з використанням CD/DVD-пристрою;

на підставі **img-образу** з використанням віртуального floppy-пристрою;

3. Зробіть аналіз функціональних можливостей, умов запуску, порядку роботи (правил гри) із програмами KolibriOS зі вказаного в табл. 2.1 варіанта завдання.

Таблиця 2.1

Індивідуальні завдання для виконання

Варіанти	Назви ігор	Demo	Оброблення даних	Системний сенсор	Команди		Обчислення
1	2	3	4	5	6		7
1	ArcanII	UnvWater	Hex-редактор	Диспетчер процесів	about	ls	$f(x) = x^2$
2	C4	"Феєрверк"	Калькулятор	PCI-пристрій	alias	mkdir	f(x) = ax

Продовження табл. 2.1

1	2	3	4	5	6		7
3	Fharach's crypt	"Павутина"	Tinypad	CPUID	cd	more	$f(x) = x^3$
4	Find Numbers	TranTest	TextEdit	GHOST-монітор	clear	ps	$f(x) = 1 / x$
5	Flood-it	Life	Table	K.bus disconnected	cp	pwd	$f(x) = x^2 + x$
6	Freecell	Moveback	Hex-редактор	HDD-інформатор	date	rm	$f(x) = x^2 + 1$
7	Just Clicks	Plasma	Калькулятор	Диспетчер процесів	echo	rmdir	$f(x) = 1 - x$
8	Memory blocks	"Труба"	Tinypad	PCI-пристрій	free	touch	$f(x) = ax + b$
9	Phenix	"Очі"	TextEdit	CPUID	history	uptime	$f(x) = x^{-2}$
10	Red Square	"Кольори"	Table	GHOST-монітор	kill	ver	$f(x) = x^2 + a$
11	Rocket Forces	"Фрактал"	Hex-редактор	K.bus dis-connected	ls	cd	$f(x) = a / x$
12	SQ_Game	"Кружок"	Калькулятор	HDD-інформатор	mkdir	alias	$f(x) = x^4$
13	Xonix	"Шестерні"	Tinypad	Диспетчер процесів	more	about	$f(x) = ax / b$
14	"Атака"	CubeLine	TextEdit	PCI-пристрій	ps	clear	$f(x) = 1 – 1 / x$
15	"Гомоку"	"Серце"	Table	CPUID	pwd	more	$f(x) = 1 / x^{-3}$
16	"Змійка"	"3D-лабіринт"	Hex-редактор	GHOST-монітор	rm	mkdir	$f(x) = x^2 + 1$
17	"Косарка"	UnvWater	Калькулятор	K.bus disconnected	rmdir	date	$f(x) = 1 - x$
18	"Лабіринт"	"Феєрверк"	Tinypad	HDD-інформатор	touch	pwd	$f(x) = ax + b$
19	"Мільйонер"	"Павутина"	TextEdit	Диспетчер процесів	uptime	kill	$f(x) = x^{-2}$
20	"Морський бій"	TranTest	Table	PCI-пристрій	ver	uptime	$f(x) = x^2 + a$
21	"Новий Понг"	Life	Hex-редактор	CPUID	alias	touch	$f(x) = a / x$
22	"Падіння"	Moveback	Калькулятор	GHOST-монітор	about	rmdir	$f(x) = x^4$

Закінчення табл. 2.1

1	2	3	4	5	6		7
23	"Понг"	Plasma	Tinypad	K.bus disconnected	cd	rm	f(x) = ax / b
24	"Квачі"	"Труба"	TextEdit	HDD-інформатор	clear	ls	f(x) = 1 – 1 / x
25	"Реверсі"	"Очі"	Table	Диспетчер процесів	cp	ver	$f(x) = 1 / x^{-3}$
26	"Сапер"	"Кольори"	Hex-редактор	PCI-пристрій	date	history	$f(x) = x^2$
27	"Судоку"	"Фрактал"	Калькулятор	CPUID	echo	free	f(x) = ax
28	"Тетрис"	"Кружок"	Tinypad	GHOST-монітор	free	echo	$f(x) = x^3$
29	"Трубопровід"	"Шестерні"	TextEdit	K.bus disconnected	history	cp	f(x) = 1 / x
30	"Шашки"	"Серце"	Table	HDD-інформатор	kill	ps	$f(x) = x^2 + x$

4. Поясніть призначення, порядок запуску, розташування на RAM-носії та початковий асемблерний код демонстраційної програми, указаної в табл. 2.1.

5. Поясніть призначення, порядок використання програми оброблення даних, що входять до складу KolibriOS, заданою в табл. 2.1. Наведіть приклад оброблення даних, підтвердивши відповідними скриншотами.

6. Зробіть аналіз системного сенсора, відповідно до індивідуального варіанта завдання, указаного в табл. 2.1. Наведіть відповідні скриншоти з поясненнями вказаних у них параметрів.

7. Визначте призначення, параметри виклику і приклад використання вбудованих типових команд інтерпретатора командного рядка, відповідно до номера індивідуального варіанта (див. табл. 2.1). Зверніть увагу, що допомогу за потрібною командою можливо отримати у форматі: `help` *`найменування команди`*.

Додаткові завдання

1. Виконайте установлення одного з вільно поширюваних компіляторів для KolibriOS.

2. Розробіть програму виконання обчислення функції f(x), заданого в табл. 2.1.

Контрольні запитання

1. Визначте мету використання ОС.
2. У чому полягають принципові відмінності ОС KolibriOS і клонів Windows?
3. Що дозволяє виявити використання служб ОС?
4. Назвіть основні принципи побудови ОС.
5. Укажіть призначення основних модулів ОС KolibriOS.
6. Назвіть основні версії KolibriOS.
7. Назвіть використовувану мову програмування в Kolibri OS.
8. Як характеризують операційну систему Kolibri OS?
9. Назвіть тип архітектури ядра Kolibri OS.
10. Назвіть підтримувану мову програмування в Kolibri OS.
11. Назвіть модуль монолітного ядра Kolibri OS.
12. Назвіть складову частину модуля управління потоками ядра Kolibri OS.
13. Що перемикає планувальник завдань ядра Kolibri OS?
14. Який тип розподілу пам'яті використовують у Kolibri OS?
15. Назвіть примітив синхронізації в Kolibri OS.
16. Назвіть стандартний пристрій, із яким взаємодіє модуль уведення-виведення ядра Kolibri OS.

Лабораторна робота 3
Дослідження операційної системи QNX NEUTRINO

Мета роботи: ознайомлення з основними принципами встановлення (інсталяції) операційної системи QNX і її конфігурації на віртуальних машинних середовищах, а також набуття практичних навичок у побудові та дослідженні QNX у процесі інсталяції.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно вивчити принципи й логіку роботи QNX-інсталятора, його основні параметри під час запуску в режимі командного рядка. Навчитися

виконувати підготовчі операції на початковому етапі встановлення QNX. Особливу увагу слід приділити процесу конфігурації ОС під час установлення і після завершення кінцевого етапу. Вивчити вимоги до апаратного забезпечення ОС.

Додаткову інформацію під час підготовки до роботи можна отримати в [27; 29; 30].

Теоретичні відомості

QNX – це POSIX-сумісна операційна система реального часу, призначена, переважно, для вбудованих систем. Уважають за одну із кращих реалізацій концепції мікроядерних операційних систем.

Як мікроядерна операційна система QNX заснована на ідеї роботи основної частини своїх компонентів як невеликих завдань, названих сервісами. Це відрізняє її від традиційних монолітних ядер, у яких ядро операційної системи – це одна велика програма, що складається з великої кількості "частин", кожна зі своїми особливостями. Використання мікроядра у QNX дозволяє користувачам (розробникам) вимкнути будь-яку непотрібну нову функціональність, не змінюючи ядро. Замість цього можна просто не запускати певний процес.

Система достатньо невелика, щоб у мінімальній комплектації вміщатися на одну дискету, разом із цим її вважають за дуже швидку і належним чином "закінчену" (що практично не містить помилок).

QNX Neutrino, випущену 2001 року, перенесено на багато платформ, і зараз вона здатна працювати практично на будь-якому сучасному процесорі, використовуваному на ринку вбудованих систем. Серед цих платформ наявні сім'ї x86, MIPS, PowerPC, а також такі спеціалізовані сім'ї процесорів, як SH-4, ARM, StrongARM і xScale.

Версія для некомерційного використання є доступною для скачування на веб-сайті розробника.

Це не BSD-ліцензія, і тим більше не GPL будь-якого роду. Ім'я їй – QNX hybrid software model. Вона нагадує ліцензію, за якою є можливість стороннім розробникам уносити зміни до коду, безкоштовна – для некомерційного використання і платна – для використання комерційного. Причому розробники не зобов'язані робити надбанням громадськості свої досягнення, а повністю можуть зберігати їх у складі закритих систем.

Є QNX Neutrino у двох варіантах – 16-розрядна версія і 32-розрядна. Остання здатна виконувати як 16- так і 32-розрядні застосування одночасно, для цього потрібно лише встановлення двох відповідних системних бібліотек, що розподіляють. Таким чином, фактично є дві операційні системи в одній упаковці. Крім того, ця версія використовує всі можливості процесорів Intel для апаратного захисту пам'яті, зокрема механізм сторінок.

Інсталяція. Найпростіший спосіб установлення QNX Neutrino з інсталяційного CD, який у вигляді образу можна завантажити із сайта виробника. Диск цей є завантажувальним, і тому ніяких додаткових маніпуляцій не потребує.

Типове установлення в розділ QNX.

Усе встановлення QNX у розділ виконують у текстовому консольному режимі. Перевага такого встановлення – це правильне встановлення, як і для всякої іншої OS. Але багато в чому переваги визначають тим, для чого необхідно використовувати систему. Установлення в розділ QNX працює з оригінальною файловою системою QNX, яка:

стійка до відмов;

істотно швидша.

Установлюють у SETUP комп'ютера первинне завантаження із CD (рис. 3.1).

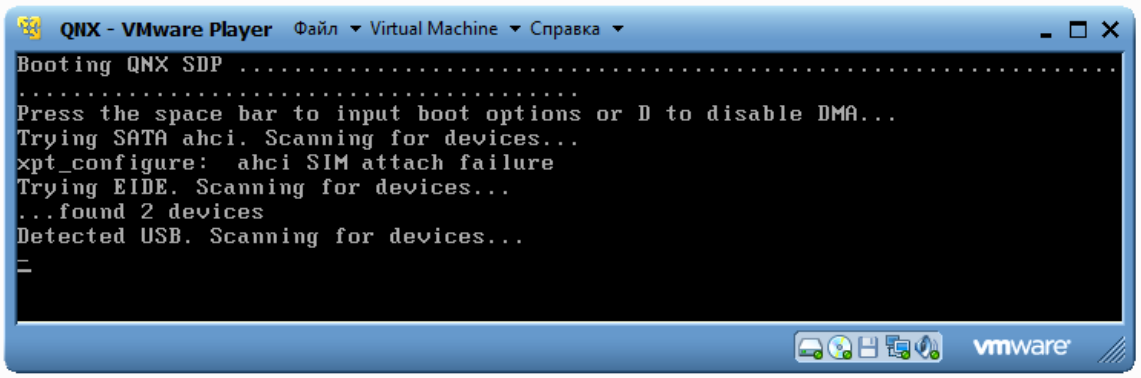


Рис. 3.1. **Процес завантаження QNX і сканування пристроїв**

Після завантаження користувачеві пропонують альтернативні можливості на вибір (названо тільки основні, реально їх більше: режим налаштування, safe режим тощо) (рис. 3.2).

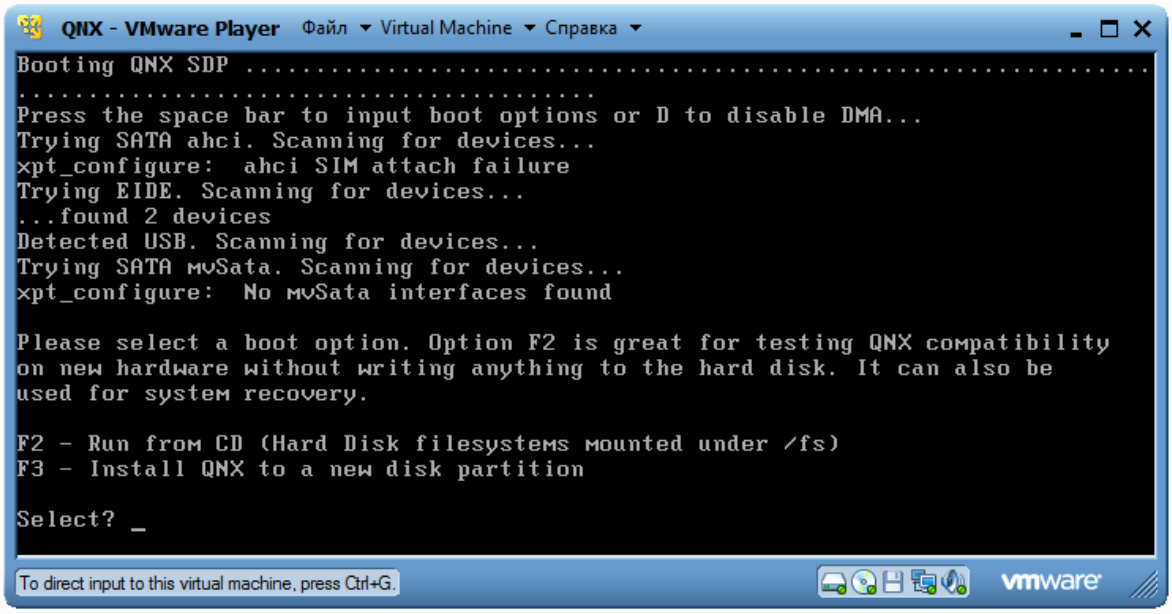


Рис. 3.2. **Вибір варіанта завантаження**

Вибір здійснюють функціональними клавішами F2 та F3 на клавіатурі. F2 – запуск із компакт-диска, F3 – установлення QNX на новий розділ жорсткого диска.

1. **F2** – повне завантаження працездатної системи з Live CD. Отримавши таким чином працездатну QNX OS, можна виконувати будь-які дії, користуючись можливостями системи, зокрема, і ручне післяопераційне встановлення нового екземпляра на диск.

2. **F3** – завантаження на HDD QNX з CD із підмонтуванням уже наявної кореневої файлової системи QNX із диска. Дозволяє таким чином працювати з раніше створеним екземпляром системи (на диску може бути більше ніж один екземпляр кореневої файлової системи). Режим створює новий розділ QNX на диску з подальшим установленням у нього системи QNX. Для того щоб скористатися цим (напівавтоматичним) режимом створення розділу і системи, потрібно передбачити на диску нерозмічений простір необхідної величини, можливо, видаливши розділ, що раніше був на ньому (наприклад, програмою fdisk Windows). Якщо буде вибрано режим створення розділу з установленням, то система запропонує використовувати під створюваний розділ:

увесь вільний простір;

половину вільного простору;

чверть вільного простору тощо.

Після цього система:

готує розділ до створення файлової системи (команда dinit);

позначає створений розділ як завантажувальний;

записує початковий завантажувач QNX;

копіює файл образу ядра .boot на розділ;

копіює кореневу файлову систему QNX на розділ.

У процесі встановлення системи користувача, так само як під час установлення в FAT32 запитають пароль користувача **root**, але можна залишити його порожнім (рис. 3.3). Додаткові реєстраційні записи користувача створювати не будуть: це ще належить зробити пізніше вручну під час адміністрування створеної системи.

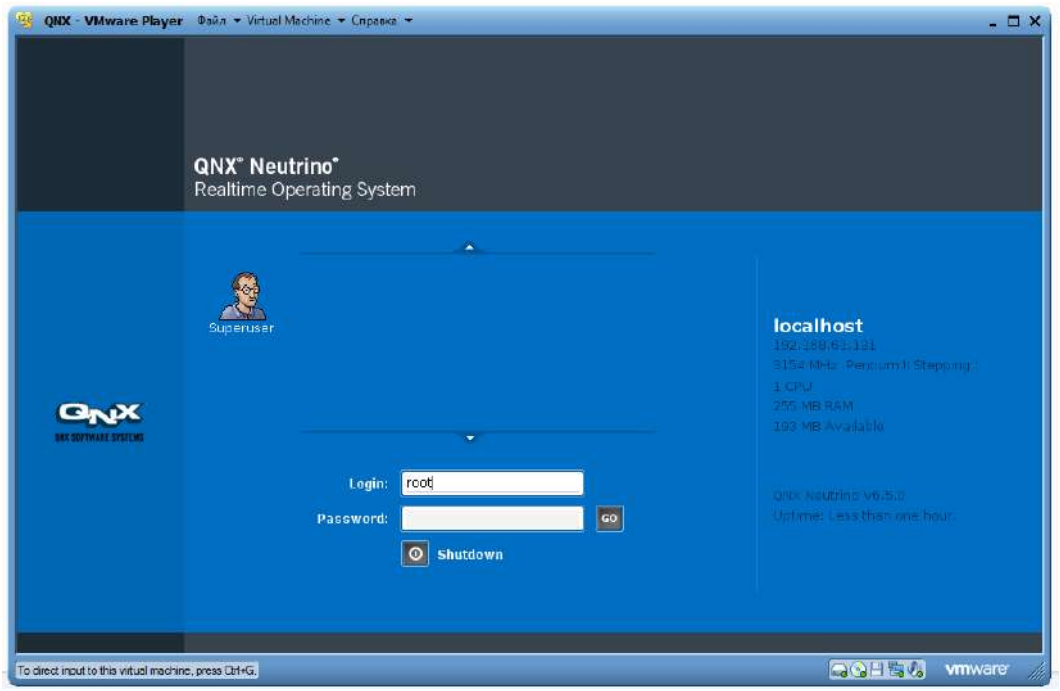


Рис. 3.3. **Вікно введення пароля root адміністратора Root**

Наступне перезавантаження роблять уже з розділу QNX під управлінням завантажувача QNX. Під час завантаження він дозволяє вибрати розділ диска, із якого буде відбуватися завантаження, тобто вибрати одну зі встановлених на комп'ютері ОС (за наявності декількох дисків завантажувач QNX дозволяє вибрати й диск завантаження).

Усе встановлення відбувається в текстовому режимі. Починається воно із пропозиції створити дискові розділи, із чим можна погодитися, натиснувши Enter, або заперечити, натиснувши Esc, після чого слідує перезавантаження. Можна також натиснути клавішу v (від англ. verbose), після

чого всі подальші кроки будуть супроводжувати докладними коментарями (рис. 3.4).

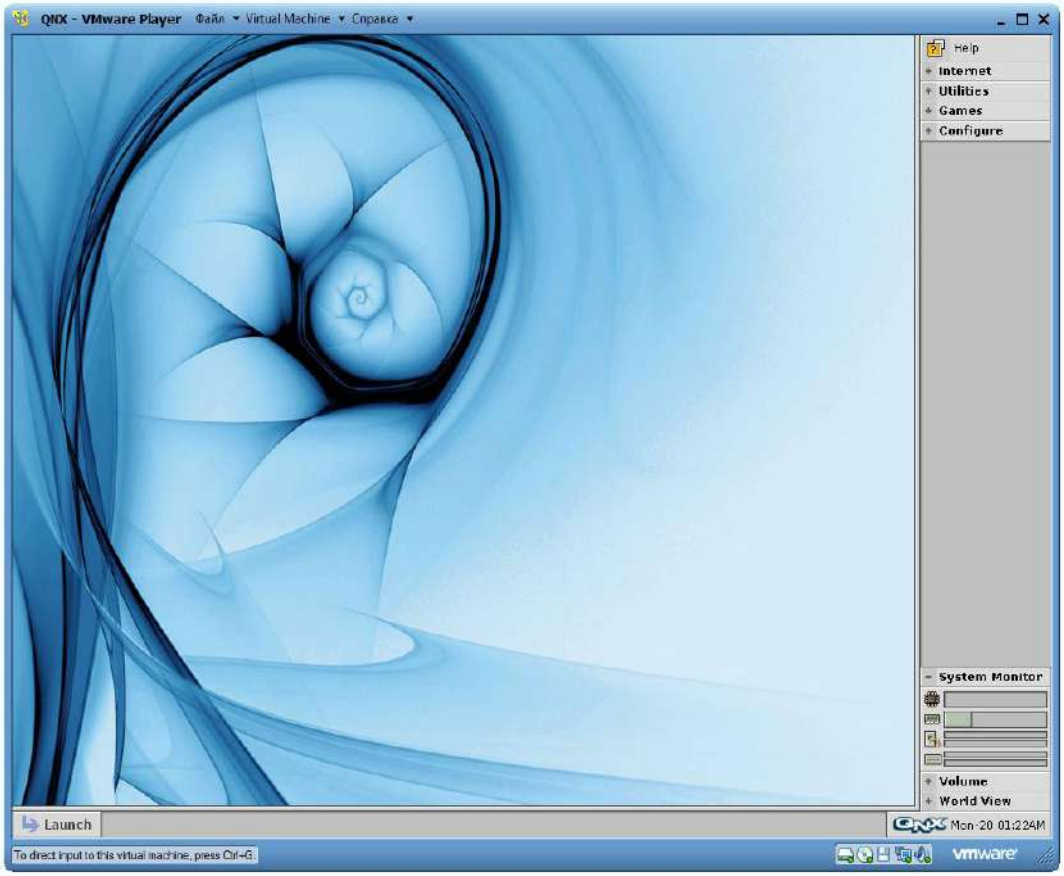


Рис. 3.4. ***Робочий стіл*** **QNX Neutrino**

Використання клавіатури в ОС QNX

Під час уведення в командному рядку можна використовувати такі клавіші:

<←> < →> < Home> <End> <Backspace> <Del> <Ins>.

Ініціація виконання команди – <**Enter**>**.**

< **Ctrl**>+<**U**> – видалення рядка.

Виклик історії команд: <↑> <↓>.

Перемикання віртуальних консолей (**/dev/con1 ...)**:

< **Ctrl**>+<**Alt**>+<**Enter**> або <**Ctrl**>+<**Alt**>+<+> – перемикання на наступну консоль;

< **Ctrl**>+<**Alt**>+<-> – перемикання на попередню консоль;

< **Ctrl**>+<**Alt>+<n>** – перемикання на консоль із номером **n**.

Перемикання шрифтів здійснюють за допомогою команди **cfont**.

Швидке перемикання шрифтів:

< **Ctrl**>+<**Alt**>+<>> – перемикання на наступний шрифт (до **n**);

< **Ctrl**>+<**Alt**>+<<> – перемикання на попередній шрифт (до **0**).

Кінець файла – <**Ctrl**>+<**D**>

Завершення процесу (термінального):

<Ctrl>+<**C>** або **<Ctrl**>+<**Break**>

Припинення виведення на екран:

<**Ctrl**>+<**S**>

Продовження виведення на екран:

<**Ctrl**>+<**D**>

Перезавантаження комп'ютера (програмна):

< **Ctrl**>+<**Alt**>+<**Shift**>+<**Del**>.

Основні команди в ОС QNX

Під час роботи користувача з ОС QNX використовують такий синтаксис команд (на прикладі команди **more**):

more [-ceisu] [-n number] [-x tabstop] [file...];

[...] – необов'язкові опції або аргументи команди;

-a | -f – альтернатива опцій;

... – можливе повторення цього аргументу.

Перенаправлення введення-виведення

Більшість команд читає своє введення з файла стандартного введення (**stdin**), яке зазвичай є клавіатурою, і записує своє виведення у стандартний файл виведення (`stdout`), який зазвичай є екраном дисплея. Проте, можливо:

читати з файла відмінного від клавіатури;

писати у файл відмінний від екрана дисплея;

читати з файла або іншого пристрою **<** (уведення перенаправлених символів);

писати у файл або в інший пристрій **>** (виведення перенаправлених символів);

перенаправлення **stdout** у файл, додавання до файла >> (додавання символів, що виводять);

канал `stdout` прямо в іншу команду **|** (конвеєр, канал символів).

Наприклад, `ps > /tmp/pr1.`

Отримання допомоги щодо конкретної команди відбувається таким чином: **use** *імя_команди.*

Наприклад, **use cp.**

Використання універсальних символів

***** – заміна будь-якого числа (**>= 0**) будь-яких символів;

? – заміна будь-якого одиночного символу;

[ ] – заміна будь-яких символів, що містяться в дужках. Може бути діапазон **[1 – 3] [a – c]**;

! – виключення символів, специфікованих у дужках **[!a]**.

Приклади:

```
cp f* /tmp
cp *d /tmp
cp freg? /tmp
cp freg[123] /tmp або cp freg[1-3] /tmp
cp *.[ch] /tmp
cp *.[!o] /tmp
```

Уведення множини команд:

```
cp freg /tmp; ls /tmp
```

Робота у файловій системі ОС QNX

Ім'я файла може бути до 48 символів у довжину.

Можливе застосування в іменах файлів символів із таких діапазонів:

```
0x00 . 0x1F
0x2f (/)
0x7F (rubout)
0xFF
```

Але для переношуваності програм бажано в іменах файлів використовувати символи, визначені в **POSIX**:

```
a...z   A...Z   0 1 ... 9   _   .   -
```

(не можна використовувати символ **"-"** як перший символ імені файла).

Суфікси в іменах файлів нічого не означають. Щодо суфіксів є локальна угода (наприклад .**txt** .**c** .**h** .**o** і тощо).

Імена шляхів (маршрути)

/ – кореневий каталог (директорій).

//2/ – кореневий каталог на 2-му вузлі.

Поточний директорій – ...

Домашній директорій (/home/slava).

Повернення в домашній каталог cd.

Типова файлова система (ФС) QNX

У файловій системі ОС QNX зазвичай наявні такі директорії (каталоги) (табл. 3.1):

Таблиця 3.1

Структура каталогів ФС ОС QNX

Каталоги	Опис
`/bin`	виконувані файли
`/boot`	makefile образу ОС
`/boot/build`	make-файли для побудови образів
`/boot/images`	файл образу ОС
`boot/sys`	системні процеси, необхідні під час завантаження
`/etc`	ініціалізації та інші файли
`/etc/config`	sysinit і конфігураційні файли
`/etc/readme`	інформаційні файли про версію програмного забезпечення
`/etc/readme/technote`	технічні зауваження
`/tmp`	локалізація, за умовчанням, тимчасових файлів
`/usr/bin`	виконувані файли
`usr/include`	файли заголовків для С компілятора
`/usr/lib`	бібліотеки для С компілятора
`/usr/lib/terminfo`	файли опису терміналів
`/usr/lib/APPLICATION`	інстальовані застосування QNX
`/usr/spool/lp`	робочі файли для системних спулерів
`/home/USERID`	домашні каталоги користувачів

Останні дві директорії конфігуруються системним користувачем.

Найбільш часто використовувані команди QNX

Робота з директоріями

cd – зміна робочої директорії;

mkdir – створення директорії;

rmdir – видалення директорії;

pwd – друкування імені робочої директорії;

ls [-l] – список змісту директорії.

Робота з файлами

diff – звіт про відмінність змісту двох файлів;

cat – конкатенація файлів, виведення вмісту файла на екран;

cp – копіювання файлів;

wc – обхід структури директорій і виконання команд;

more – показ умісту файла в буферизованому вигляді;

less – інтерактивний (керований) перегляд виведення на екран;

lp – виведення змісту файла на принтер;

mv – перенесення файла;

rm – видалення файла;

grep – пошук по рядку, заданому у вигляді шаблону;

sort – сортування, злиття або перевірка послідовності текстового файла.

Інші команди

who – виведення на екран користувачів у системі;

ps – виведення на екран звіту про статус процесів;

sin – виведення системної інформації на екран;

find – пошук файлів;

write – посилання повідомлення на термінал користувача;

wall – посилання широкомовного повідомлення.

Загальні завдання для виконання

1. Промоделюйти різні варіанти завантаження системи:

F2 – завантаження передустановленої системи з використанням iso-образа на CD/DVD-пристрою;

F3 – інсталяції системи на віртуальний жорсткий диск (буде потрібно введення ключа продукту).

2. Зробіть аналіз установленого програмного забезпечення, указаного в табл. 3.2 індивідуального завдання.

Таблиця 3.2

Індивідуальні завдання для виконання

Варіанти	Software	Команди		Обчислення
1	2	3		4
1	Web Browser	cat	uncompress	Сума позитивних елементів масиву
2	Video Pocker	chgrp	sloginfo	Сума елементів масиву, розташованих між першим і останнім негативними елементами
3	Text Editor	chmod	shutdown	Добуток елементів масиву, розташованих між максимальним і мінімальним елементами
4	Terminal charset setup	chown	uname	Максимальний елемент масиву
5	Terminal	confstr	umount	Сума елементів масиву з непарними номерами
6	Solitaire	pwd	uesh	Сума елементів масиву, розташованих між першим і останнім позитивними елементами
7	Snapshot	sh	su	Номер максимального елемента масиву
8	Regiion View	csplit	stty	Сума модулів елементів масиву, розташованих після мінімального за модулем елемента
9	Print Manager	dd	split	Сума елементів масиву, розташованих після першого позитивного елемента

Продовження табл. 3.2

1	2	3		4
10	Othello	dispconf	netmanager	Сума елементів масиву, розташованих між першим і другим негативними елементами
11	Peg	df	sloginfo	Сума негативних елементів масиву
12	Network	du	sh	Номер мінімального за модулем елемента масиву
13	Mouse	mv	zcat	Мінімальний елемент масиву
14	Localization	ed	slay	Добуток елементів масиву, розташованих між максимальним за модулем і мінімальним за модулем елементами
15	Image Viewer	esh	mkdir	Кількість елементів масиву, більших ніж число K
16	Find	ex	more	Максимальний за модулем елемент масиву
17	File Manager	false	true	Добуток елементів масиву, розташованих після максимального за модулем елемента
18	Display	fesh	chown	Сума модулів елементів масиву, розташованих після першого негативного елемента
19	Dialer	shutdown	gzip	Номер мінімального елемента масиву
20	Columns	hostname	who	Кількість негативних елементів масиву
21	Capture Screen	su	chmod	Добуток елементів масиву, розташованих між першим і другим нульовими елементами
22	Calculator	ksh	df	Сума цілих частин елементів масиву, розташованих після останнього негативного елемента
23	Terminal	cpio	slay	Сума позитивних елементів масиву, розташованих до максимального елемента
24	Solitaire	ls	pax	Кількість елементів масиву, менших ніж число K
25	Snapshot	mkdir	pidin	Сума елементів масиву, розташованих до мінімального елемента
26	Regiion View	cp	ps	Сума елементів масиву, розташованих до останнього позитивного елемента

Закінчення табл. 3.2

1	2	3		4
27	Peg	mount	rm	Добуток позитивних елементів масиву
28	Print Manager	more	pwd	Сума елементів масиву, розташованих між першим і другим позитивними елементами
29	Othello	mv	script	Номер максимального за модулем елемента масиву
30	Network	on	sendnto	Добуток негативних елементів масиву

3. Визначте призначення, параметри виклику і приклад використання вбудованих типових команд інтерпретатора терміналу, відповідно до номера індивідуального варіанта (див. табл. 3.2).

4. Проведіть дослідження зареєстрованих у системі пристроїв. Результати занесіть у табл. 3.3.

Таблиця 3.3

Зареєстровані в системі пристрої

№ п/п	Назви пристрою	Значення статусу	Примітки
1	CD-ROM		
2	Клавіатура		
3	Мишка		
4	Мережева плата		
5	Звуковий драйвер		
6	Дисплей		
7	Принтер		

Результати встановлення відобразити у звітах зі вказівкою алгоритму встановлення, його особливостей і описом послідовності дій під час виконанні кожного етапу завдань.

Додаткове завдання

Розробіть у QNX Photon Application Builder програму виконання в одновимірному масиві, що складається з n речових елементів, обчислення, задане в табл. 3.2.

Контрольні запитання

1. Визначте мету використання ОС.
2. У чому полягають принципові відмінності QNX?
3. Що дозволяє виявити використання служб ОС?
4. Назвіть основні принципи побудови ОС.
5. Укажіть призначення основних модулів QNX Neutrino.
6. Назвіть основні версії QNX і місця їхнього використання.
7. Назвіть тип ядра в QNX.
8. Який вид програмування підтримується в QNX Neutrino?
9. Назвіть архітектуру QNX Neutrino.
10. Укажіть складову частину архітектури QNX Neutrino, що утворює частину її ядра.
11. Укажіть складову частину архітектури QNX Neutrino, яка управляє певним видом ресурсів.
12. Назвіть основний структурний елемент підсистеми QNX Neutrino.
13. Який розмір системного розділу в QNX Neutrino?
14. Який розмір призначеного для користувача розділу у QNX Neutrino?
15. Де в пам'яті зберігають коди додатка в QNX Neutrino?
16. Що утворюють структурні елементи призначеного для користувача розділу у QNX Neutrino?
17. Яку трансляцію адрес використовують для блоків у QNX Neutrino?
18. Що становлять програмні коди в архітектурі QNX Neutrino?
19. Який диспетчер виконує відстежування завантаження процесів через задані інтервали часу у QNX Neutrino?
20. Що має потік у QNX Neutrino?
21. Який диспетчер QNX Neutrino забезпечує для додатків обмін подіями, що не відповідають протоколу TCP/IP?
22. Назвіть тип архітектури QNX Neutrino.
23. Який адресний простір підтримує *Диспетчер пам'яті* у QNX Neutrino?
24. Чим дозволяє обмінюватися між собою завданням QNX Neutrino?
25. Чим управляє "ядро в ядрі" QNX Neutrino?
26. Який вид становить файлова систем QNX Neutrino?
27. Назвіть основні структурні елементи команди в QNX Neutrino.

Розділ 2. Оперативна пам'ять, потоки та процеси

Лабораторна робота 4
Моделювання процесів в операційній системі

Мета роботи: ознайомлення з основними принципами моделювання і дослідження функціонування операційних систем (ОС), набуття практичних навичок у моделюванні процесів в ОС, прагнучи досягнення її максимальної продуктивності.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно вивчити алгоритми планування виконання процесів на процесорі, стратегії заміни сторінок. Навчитися виконувати обчислення в табличному процесорі Excel. Особливу увагу приділити питанням синхронізації процесів із використанням механізму семафорів.

Додаткову інформацію під час підготовки до роботи можна отримати в [5; 6].

Теоретичні відомості

У лабораторній роботі моделюють деякі аспекти функціонування ОС під час оброблення призначених для користувача та системних процесів. У роботі використовують програмну модель, що відображає такі аспекти функціонування ОС:

подання процесу в системі;

взаємодія процесів у системі;

виділення сторінок пам'яті за запитами за допомогою використання алгоритмів заміщення сторінок;

розподіл ресурсів обчислювальної системи між кількома процесами за допомогою механізму семафорів.

Модель має такі параметри, значення яких можна міняти, залежно від заданого варіанта:

1. Три фіксовані пріоритети процесів – користувачеві, серверні та системні завдання (завдання ядра).

2. Максимальна кількість призначених для користувача процесів – 10.

3. Кількість серверних процесів операційної системи – 9, серед них:

Диспетчер пам'яті, що здійснює динамічне перетворення адрес;

Диспетчер файлової системи (ФС), обслуговуючий відповідні системні виклики.

4. Кількість системних завдань – 7, серед них:

завдання, обслуговуюче принтер;

завдання, обслуговуюче термінал;

завдання, обслуговуюче жорсткий диск (вінчестер);

завдання, обслуговуюче flash-диск (дисковід);

системний годинник;

завдання, обслуговуюче системні виклики (System Task) (для простоти взаємодії призначених для користувача процесів із ядром здійснюють без її участі);

процес апаратури, "порожній" процес, що відбувається під час простою процесора.

Взаємодія процесів у системі відбувається за допомогою механізму повідомлень. Процес, що послав повідомлення, стає блокованим. Процес, що отримав повідомлення, – готовим до виконання. Системні завдання мають найбільший пріоритет, потім ідуть, відповідно, серверні та призначені для користувача процеси.

Планувальник за замовчуванням використовує дисципліну планування rabbit ring (RR), водночас процеси різних пріоритетів утворюють різні циклічні черги, і диспетчер кожного разу вибирає найбільш пріоритетний процес.

Усі процеси мають віртуальний адресний простір від 0 до деякого N (старша віртуальна адреса), довільним чином розбитий на сегменти коду, даних і стека.

Розподіл ресурсів у системі відбувається за допомогою семафорів, причому термінал і принтер процес використовує неподільно, але всі процеси можуть спільно використовувати flash-дисковід і жорсткий диск.

4.1. Управління процесами

Поточний стан віртуального комп'ютера, наданий користувачеві, називають **образом (image)**. Процес – це виконання образу. Образ складається з образу пам'яті, значень загальних регістрів, стану відкритих файлів, поточної директорії та іншої інформації. Образ процесу під час виконання розміщено в основній пам'яті. Образ може бути збережено на диску,

якщо будь-якому більш пріоритетному процесу буде потрібно місце в основній пам'яті. Образ пам'яті складається із трьох сегментів (рис. 4.1):

Сегмент реентерабельних процедур
Сегмент даних
Сегмент стека

Рис. 4.1. **Образ пам'яті**

Сегмент реентерабельних процедур може спільно використовуватися декількома процедурами: у первинній пам'яті такі сегменти, які розподіляють, подано єдиною копією. Сегментами реентерабельних процедур система управляє централізовано за допомогою таблиці процедур (рис. 4.2).

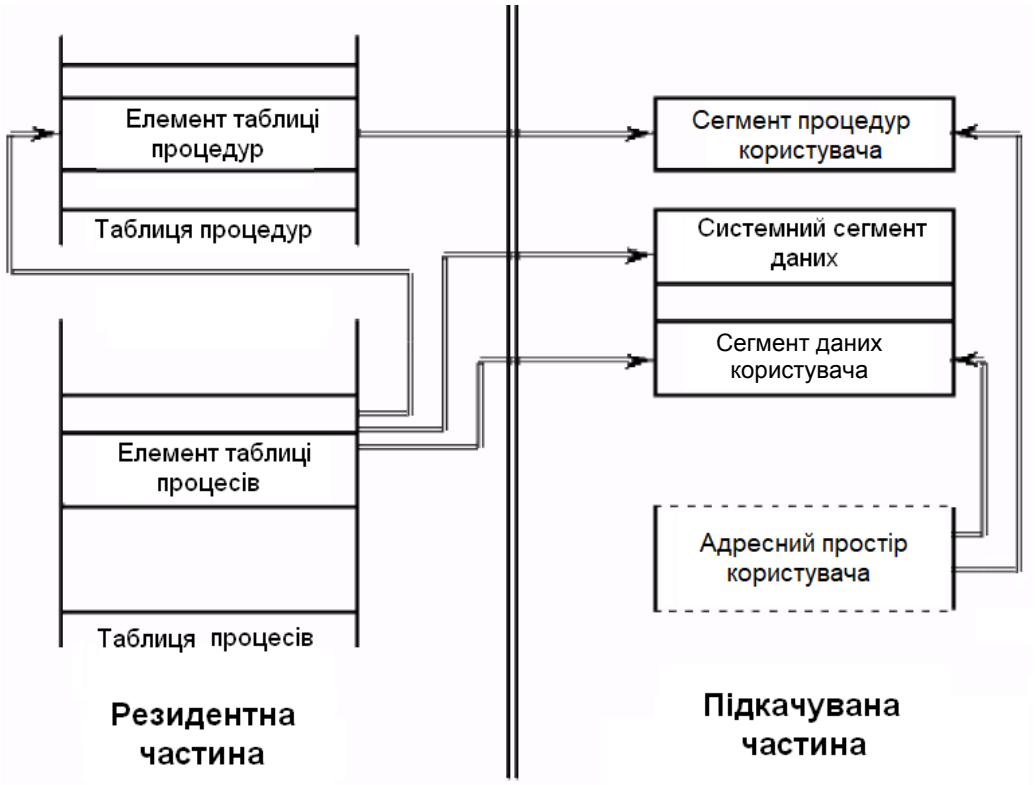


Рис. 4.2. **Таблиця управління процесом**

Кожен елемент таблиці містить адреси сегмента як у первинній, так і вторинній пам'яті. Коли процес перший раз звертається до розподіленого сегмента, цей сегмент поміщається у вторинну пам'ять і заводиться елемент у таблиці процедур із відповідними адресами та лічильником, який указує на кількість процесів, що спільно використовують цей сегмент.

Коли в лічильнику опиниться нуль, звільняється елемент таблиці й області первинної та вторинної пам'яті.

Сегмент даних розташовано слідом за сегментом процедур, що може зростати вниз. Він містить дані, записувані та зчитувані тільки одним процесом. Системні дані, що належать до процесу, зберігають в окремому сегменті фіксованого розміру. Цей системний сегмент закінчується разом із процесом. Він містить:

стан регістрів;

дескриптори (описувачі) відкритих файлів;

дані для розрахунків за використання машини;

область робочих даних;

стік для системної фази виконання процесу.

Сам процес до цього сегмента адресувати не можна. Коли процес не активний, система зберігає інформацію про процес в таблиці процесів. У ній міститься ім'я процесу, розташування його сегментів та інформація для планувальника.

Сегмент стека (див. рис. 4.1) починається зі старшої адреси у віртуальному адресному просторі та зростає вгору в міру занесення в нього інформації під час викликів підпрограм і переривань.

4.2. Синхронізація процесів

Синхронізацію процесів здійснює механізм подій. Процеси чекають подій. Події подають адресами відповідних таблиць. Батьківський процес, який чекає завершення одного з дочірніх процесів, чекає події, поданою адресою його власного елемента таблиці свого батьківського процесу.

4.3. Планування процесів

Процеси виконують у двох станах: або системному, або призначеному для користувача. У *призначеному для користувача* стані процес має доступ до призначеного для користувача сегмента даних, а в *системному* – до системного сегмента. Головною метою планування процесів у системі є забезпечення високої реактивності для інтерактивних користувачів. Планування процесів відбувається, відповідно до їхніх пріоритетів. Вищий пріоритет віддано процесу, який обробляє події й очікування. Події, пов'язані з роботою дисків, дістають наступний за старшинством пріоритет. Події, пов'язані з терміналами, часом доби та призначеними для користувача процесами, дістають, відповідно, нижчі пріоритети. Усі системні процеси мають вищий пріоритет. Призначені для користувача

пріоритети залежать від часу завантаження процесора. Чим більше процес отримує призначеного для користувача часу, тим нижчий його пріоритет. Такий метод планування забезпечує хороший час реакції під час діалогу.

4.4. Уштовхування-виштовхування

Процес може виштовхуватися у вторинну пам'ять і вштовхуватися зворотно, водночас застосовують стратегію "перший відповідний". Якщо процес запрошує додаткову пам'ять, то йому виділять нову секцію пам'яті достатнього розміру і туди копіюють уміст старої пам'яті. Стару пам'ять звільняють. Якщо в момент розширення процесу в первинній пам'яті не виявлено в наявності вільного блока, процесу відводять потрібне місце у вторинній пам'яті, його виштовхують і вштовхують знов у первинну пам'ять (уже враховуючи його новий розмір), коли в ній з'явиться достатній за розмірами вільний блок. Уштовхуванням і виштовхуванням управляє спеціальний процес підкачування.

4.5. Конфігурація обчислювальної системи

Будь-яку обчислювальну систему можна охарактеризувати, описавши її елементи, а саме: центральний процесор (ЦП), оперативну пам'ять (ОП) і пристрої введення/виведення.

Центральний процесор – це будь-який, здатний забезпечити одночасне виконання декількох процесів у реальному часі. *Оперативна пам'ять* має двопортову організацію для забезпечення паралельної роботи ЦП і каналів уведення/виведення. Пристрої введення/виведення (ПВВ) становлять необхідний мінімальний набір:

термінал (TTY);

жорсткий диск (Winchester);

дисковід (Floppy);

принтер (Printer).

Модель у своєму функціонуванні має саме таку конфігурацію.

4.6. Стан процесу

Призначений для користувача процес у цій моделі може перебувати в одному із трьох станів:

активний – йому в цей момент надано ЦП;

готовий (ready) – не володіє ЦП, але в будь-який момент часу може стати активним;

блокований (unready) – не володіє ЦП і не може стати активним, оскільки чекає події (наприклад, завершення операції введення/виведення). Під час настання цієї події процес переходить в один із перших двох станів. На рис. 4.3 показано перехід з одного стану в інший.

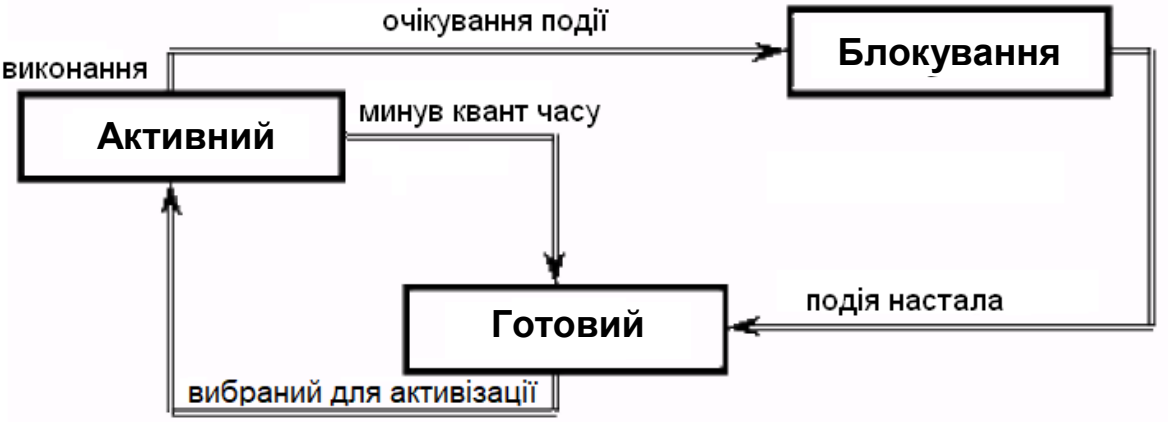


Рис. 4.3. **Стани процесу**

4.7. Взаємодія процесів у системі

У цій моделі призначені для користувача процеси взаємодіють із системними завданнями не безпосередньо, а через звернення до серверних процесів (*Memory Manager & File System*). Ці два процеси є незалежними від ядра ОС (настільки, що можливі декілька варіантів реалізації цих процесів). Їхні відмінності від призначених для користувача процесів полягають у такому:

серверні процеси мають вищий пріоритет;

серверні процеси можуть безпосередньо взаємодіяти із системами;

працюють із реальним адресним простором.

Така ієрархічна структура дозволяє реалізувати різні способи захисту як призначених для користувача програм один від одного, так і ОС від несанкціонованого доступу.

Оскільки Memory Manager і File System виконують у цій моделі виключно важливу і складну роль, доцільно розглянути їх докладніше.

4.8. Управління пам'яттю (Memory Manager)

У моделі Memory Manager виконує функцію динамічного перетворення адрес під час адресації віртуальної пам'яті (у UNIX трансляцію адрес здійснює безпосередньо ядро, а Memory Manager контролює динамічне використання пам'яті). Під час виконання Memory Manager на екрані

з'являється вікно з картою ОП. Під час виникнення сторінкової відмови Memory Manager здійснює завантаження необхідної сторінки, використовуючи поточну стратегію заміщення (вибирає користувач). Можлива також динамічна зміна розміру ОП у процесі моделювання.

4.9. Управління файлами (File System). Семафорні операції

File System містить двійкові семафори (S) для забезпечення взаємовиключення процесів під час використання пристроїв введення/виведення. Коли процес звертається до якого-небудь пристрою введення/виведення, він виконує операцію P над відповідним семафором P(S) (операція P зменшує S на 1). Якщо стан семафора дорівнює 1, то його скидають, а процес дістає доступ до ПВВ, інакше процес переводять у стан очікування (блокують).

Коли процес звільняє ресурс (наприклад, під час завершення операції введення/виведення), то, якщо є очікуючі ресурс процеси, один із них дістає доступ до ресурсу, інакше (якщо їх немає), семафор відновлює свій початковий стан (операція V(S)). Операція V збільшує S на 1.

Тимчасову діаграму виконання P, V операцій над семафорами (S) наведено на рис. 4.4, де A, B, C – позначено критичну ділянку (*CS*). Під CS мають на увазі частину процесу, протягом якого деякий загальний ресурс мають монополізувати.

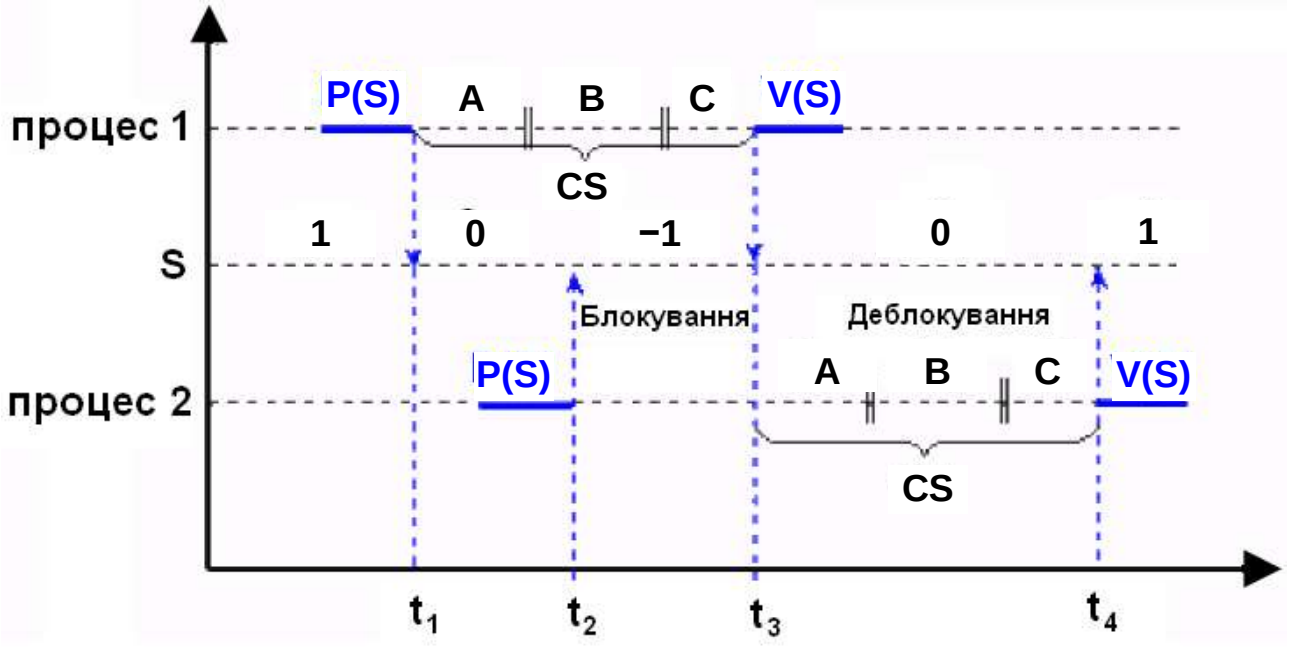


Рис. 4.4. **Тимчасова діаграма двійкового семафора**

Виконання операцій P і V у цій моделі супроводжено появою на екрані відповідних вікон. Крім того, коли виконують процес файлової системи, на екрані з'являється вікно, що зображає список ПВВ обчислювальної системи та список процесів, що очікують до них доступу.

Оскільки деякі ресурси використовуються процесами нерозподільно, то в моделі можливе виникнення тупиків (дедлоків), що відображає реальну проблему, яку необхідно вирішувати під час проектування ОС.

Під час появи вікна, що описує стан семафора, робота системи зупиняється. Для продовження роботи потрібно натиснути будь-яку клавішу.

4.10. Призначені для користувача процеси

Типи системних викликів, які генерують призначеними для користувача процесами, вибирають випадковим чином за допомогою генератора випадкових чисел.

Порядок використання моделі

1. Мета

Ця модель дозволяє вивчити деякі реальні механізми, використовувані в багатьох операційних системах, а саме:

розподіл ресурсів обчислювальної системи між декількома процесами за допомогою семафорів;

взаємодію системних і прикладних завдань;

подання процесу в системі (із погляду ОС і самого процесу);

різні рівні планування.

Модель дозволяє вивчити також залежність продуктивності обчислювальної системи від ряду параметрів ОС, таких як:

коефіцієнт мультипрограмування;

розмір кванта часу;

стратегія заміщення сторінок під час використання механізму віртуальної пам'яті.

Досягнувши оптимального поєднання цих параметрів, можна здобути максимальну продуктивність системи.

2. Порядок використання моделі

Для функціонування моделі необхідні:

персональний комп'ютер;

наявність кольорового монітора;

операційна система Windows;

достатнє для створення вихідного файла місце на диску (це значення залежить від часу моделювання і становить у середньому 200 – 300 KB).

необхідний файл моделі – **winmos.exe.**

3. Загальна організація модельованої обчислювальної системи

У ході функціонування моделі в системі виконують процеси трьох класів, що мають фіксований пріоритетом: системні завдання (завдання ядра), серверні та призначені для користувача процеси. Виділення процесам ресурсу ЦП здійснюють шляхом формування циклічних черг (*дисципліна RR*) запитів. Водночас процеси різних класів утворюють різні черги, а планувальник кожного разу вибирає найбільш пріоритетний процес. Процеси в ході роботи формують запити до ресурсів. Взаємовиключення процесів під час звернення до ресурсів здійснюється за допомогою механізму семафорів. Для роботи процесам також необхідний ресурс оперативної пам'яті. Оперативна пам'ять у моделі має сторінкову організацію. У разі браку первинної пам'яті для розміщення всіх процесів розподіл відбувається шляхом виштовхування невикористовуваних сторінок у вторинну пам'ять.

Завдання ядра. Процеси цього класу є складовою частиною ядра системи і мають найвищий пріоритет. Ця модель охоплює такі завдання ядра:

завдання, що обслуговують ресурси (*Принтер, Термінал, Вінчестер, Дисковід*). Коли відбувається зайняття або звільнення якого-небудь ресурсу система викликає відповідне обслуговче завдання. Тривалість роботи обслуговчих завдань становить один квант модельного часу;

системний годинник. Це завдання, що підтримує системний час. Система викликає це завдання кожні 10 квантів модельного часу. Тривалість роботи системного годинника становить один квант;

System Task. Це завдання в реальній ОС обслуговує системні виклики, тобто здійснює взаємодію між призначеними для користувача процесами та ядром. Проте в цій моделі для спрощення взаємодію здійснюють без її участі;

процес апаратури. Цей процес відбувається під час простою процесора.

Серверні процеси. Ці процеси є посередниками, що здійснюють взаємодію між призначеними для користувача процесами та завданнями ядра. Ці процеси також, як і призначені для користувача, незалежні від ядра системи, мають переваги перед призначеними для користувача процесами. Серверні процеси можуть безпосередньо взаємодіяти із системою і працюють із реальним адресним простором. Серверні процеси мають вищий пріоритет, ніж призначені для користувача. У цей моделі до серверних процесів належать **Диспетчер пам'яті**, що здійснює вштовхування/виштовхування сторінок пам'яті, та **Файлова система**, обслуговуючи доступ до ресурсів. Слід розглянути їх докладніше.

Диспетчер пам'яті. Якщо об'єм оперативної пам'яті, необхідної процесам більше, ніж наявної в обчислювальній системі, необхідно мати механізми логічного розширення фізичної пам'яті. У цій моделі реалізовано виділення сторінок процесам за запитами. Призначені для користувача програми працюють із логічною пам'яттю, яка відображається на фізичну пам'ять посторінково. Фізичну пам'ять розподілено на сторінки однакового розміру, і кожній зі сторінок може відповідати будь-яка сторінка логічної пам'яті. Фізично процес не обов'язково працює з безперервним блоком пам'яті, проте логічно сторінки пам'яті, із якими працює процес, залишаються суміжними. Тобто така форма заміщення не робить впливу на адресний простір програм. Логічні сторінки процесів, що не помістилися в основну пам'ять, перебувають у вторинній пам'яті, що має, переважно, значно більший об'єм і меншу швидкодію. Спочатку в оперативну пам'ять завантажують тільки перші сторінки процесів. Решта всіх сторінок завантажують згодом за запитами. У цій моделі використано стратегію глобального витіснення, тобто будь-якій сторінці фізичної пам'яті може відповідати будь-яка сторінка будь-якого завдання.

Сторінковою відмовою називають ситуацію, коли процес звертається до сторінки, що перебуває у вторинній пам'яті. У цьому разі виконується завантаження цієї сторінки в основну оперативну пам'ять. Якщо оперативну пам'ять заповнено, необхідно вивантажити одну зі сторінок у зовнішню пам'ять. Замінювану сторінку вибирають за одним з алгоритмів:

FIFO (First In, First Out, першим прийшов – першим пішов) – замінюється сторінка, раніше за всіх завантажена в оперативну пам'ять.

LFU (Least Frequently Used, найменш часто використовувана) – замінюється сторінка, до якої за час її перебування в ОП було найменше звернень.

NUR (Not Used Recently, давно не використовувана) – еталонна стратегія заміщення, заснована на прогнозі запитів процесів. Відбувається заміщення сторінки пам'яті, до якої найдовше не буде звернень.

Диспетчер пам'яті в цій моделі викликають у разі виникнення сторінкової відмови, він робить заміну сторінки, згідно з вибраним алгоритмом. Тривалість **заміщення** однієї сторінки становить п'ять квантів модельного часу.

Файлова система. У ході роботи призначені для користувача процеси формують запити до ресурсів. Для взаємовиключення в разі доступу до ресурсів у цій моделі файлова система використовує **двійкові семафори** (у реальних ОС семафорні операції виконуються безпосередньо ядром). Процес файлової системи запускають на два кванти часу кожного разу, коли який-небудь процес виконує звернення до ресурсу. Кожен ресурс має свій семафор S, що характеризує його доступність таким чином:

S = 1 – ресурс вільний;

S = 0 – ресурс зайнятий процесом;

S = −1 – ресурс зайнятий процесом і в системі є процес, блокований в очікуванні доступу до ресурсу (файлова система містить для цієї мети черги процесів, що очікують ресурсів).

Якщо процес звертається до якого-небудь ресурсу, виконується операція P(S) над відповідним семафором, зменшуючи S на 1. Якщо значення S = 1, процес дістає доступ до ресурсу, інакше – його переводять у стан очікування.

Під час звільнення процесом ресурсу виконується операція V(S). Якщо є процес, що очікує звільнення ресурсу, то він дістає доступ до ресурсу, інакше семафор установлюють в 1.

Принтер і термінал є ресурсами, що не розподіляють. Такий ресурс процес займає на тривалий час, водночас використовує процесор і може займати інші ресурси. *Дисковід* і *Вінчестер* – ресурси, що не розподіляють. Процес займає їх на короткий час, і під час роботи з ними не звертається до процесора та інших ресурсів. Наявність **у системі ресурсів, що не розподіляють, робить можливим виникнення тупикової ситуації.**

Призначені для користувача процеси. У цій моделі може існувати до 10 призначених для користувача процесів. **Призначені для користувача процеси** – це деякі програми, що не стосуються системи та запускаються користувачем із певною метою (прикладні програми). Ці процеси мають найменший пріоритет.

Адресний простір кожного процесу складається із **трьох сегментів:** сегмент коду, сегмент даних і сегмент стека. Кожен призначений для користувача процес характеризується своїм унікальним описувачем – **дескриптором.**

Дескриптор містить такі поля:

PC – покажчик на операцію, що виконується процесом у цей момент;

SP – покажчик на вершину стека;

DP – покажчик на дані;

CodeSize – розмір сегмента коду;

DataSize – розмір сегмента даних;

StackSize – розмір сегмента стека;

У ході роботи процесу значення PC і SP змінюються.

Призначені для користувача процеси за допомогою генератора випадкових чисел формують системні виклики: запити до пам'яті та ресурсів. У завдання ОС входить обслуговування цих запитів.

Загальне завдання для виконання

Робота з моделлю. Під час запуску моделі на екрані з'являється заставка. Для початку роботи необхідно в головному меню вибрати пункт *Моделювання → Моделювання...* Водночас на екрані з'являється діалог параметрів моделювання.

Діалог параметрів моделювання зображено на рис. 4.5.

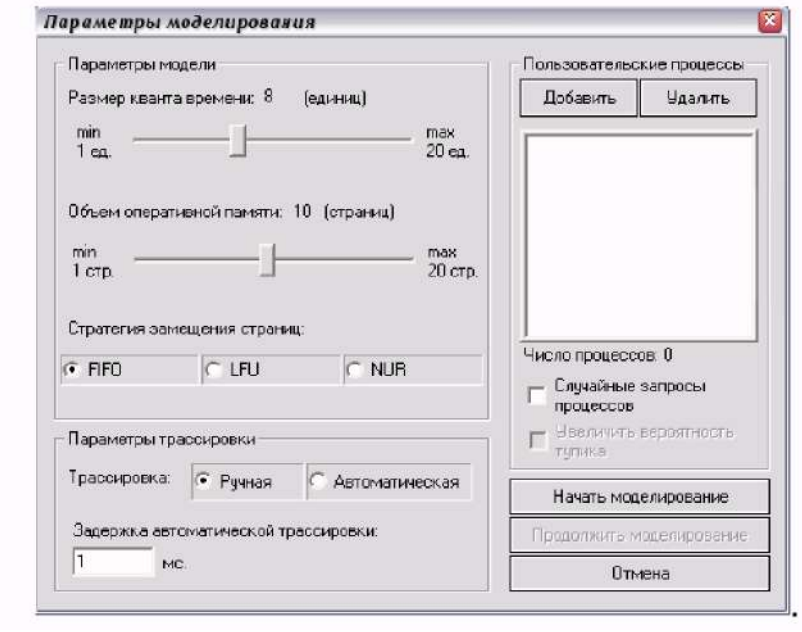


Рис. 4.5. **Діалог параметрів моделювання**

За допомогою діалогу можна виконувати такі дії:

1. Зміну параметрів функціонування моделі:

розміру кванта часу – це максимальний час, що відводиться процесу на використання CPU. Після закінчення кванта часу CPU віддається іншому процесу. Мінімальне значення – 1. Значення за замовчуванням – 8. Максимальне значення – 20;

об'єму оперативної пам'яті, його вимірюють у сторінках. Мінімальне значення – 1. Значення за замовчуванням – 10. Максимальне значення – 20;

стратегію заміщення сторінок – це алгоритм, використовуваний менеджером пам'яті для вибору сторінки, що заміщають. Доступні значення: FIFO, LFU, NUR. Значення за замовчуванням – FIFO.

2. Зміну параметрів трасування:

виду трасування. У разі ручного трасування робота системи виконується по кроках. Для просування модельного часу на одиницю вперед користувач має натиснути клавішу "Пробіл". У разі автоматичного трасування зупинка моделювання відбувається тільки за виникнення семафорної операції. Значення за замовчуванням – ручна;

затримки автоматичного трасування – інтервал часу в мс, через який відбувається збільшення модельного часу в разі автоматичного трасування. Мінімальне значення – 1. Значення за замовчуванням – 1. Максимальне значення – 5 000.

3. Призначені для користувача процеси:

додавання процесу. Для додавання призначеного для користувача процесу в систему необхідно натиснути кнопку "Додати". Під час натиснення цієї кнопки з'являється діалог, що дозволяє задати ім'я процесу, який додають. За замовчуванням процесам дають імена виду "Процес X", де X – його порядковий номер;

видалення процесу. Для видалення процесу необхідно виділити його у списку процесів і натиснути кнопку "Видалити";

випадкові запити процесів. Під час установлення цього прапорця процеси наново будуть формувати випадковим чином список запитів до системи за кожного перезапуску моделювання. У разі зняття прапорця список сформується тільки один раз під час запуску програми, і процеси кожного разу будуть формувати одні й ті самі запити. Це дозволяє ставити чисті експерименти, не роблячи знижку на випадковість запитів процесів. За замовчуванням прапорець знято. Перевірити подібність запитів процесів

можна багато разів, запускаючи моделювання та спостерігаючи стан системи у фіксований момент часу (стан системи має бути однаковим);

збільшити вірогідність тупика. Під час установлення цього прапорця список запитів процесів буде сформовано таким чином, що з великою вірогідністю в системі виникне тупикова ситуація. За замовчуванням прапорець знято.

4. Почати моделювання / продовжити моделювання / відміна:

почати моделювання. Під час натиснення цієї кнопки запускається процес моделювання з нульового моменту часу.

продовжити моделювання. Під час натиснення цієї кнопки моделювання продовжується з точки припинення. Таким чином, можлива динамічна зміна параметрів моделювання. У реальних системах параметри не змінюються динамічно, і ця можливість має суто академічний характер;

відміна. Під час натиснення цієї кнопки відбувається закриття діалогу без застосування зроблених змін.

Після натиснення кнопки "Почати моделювання" система переходить у режим моделювання.

Режим моделювання. Зовнішній вигляд програми в режимі моделювання зображено на рис. 4.6.

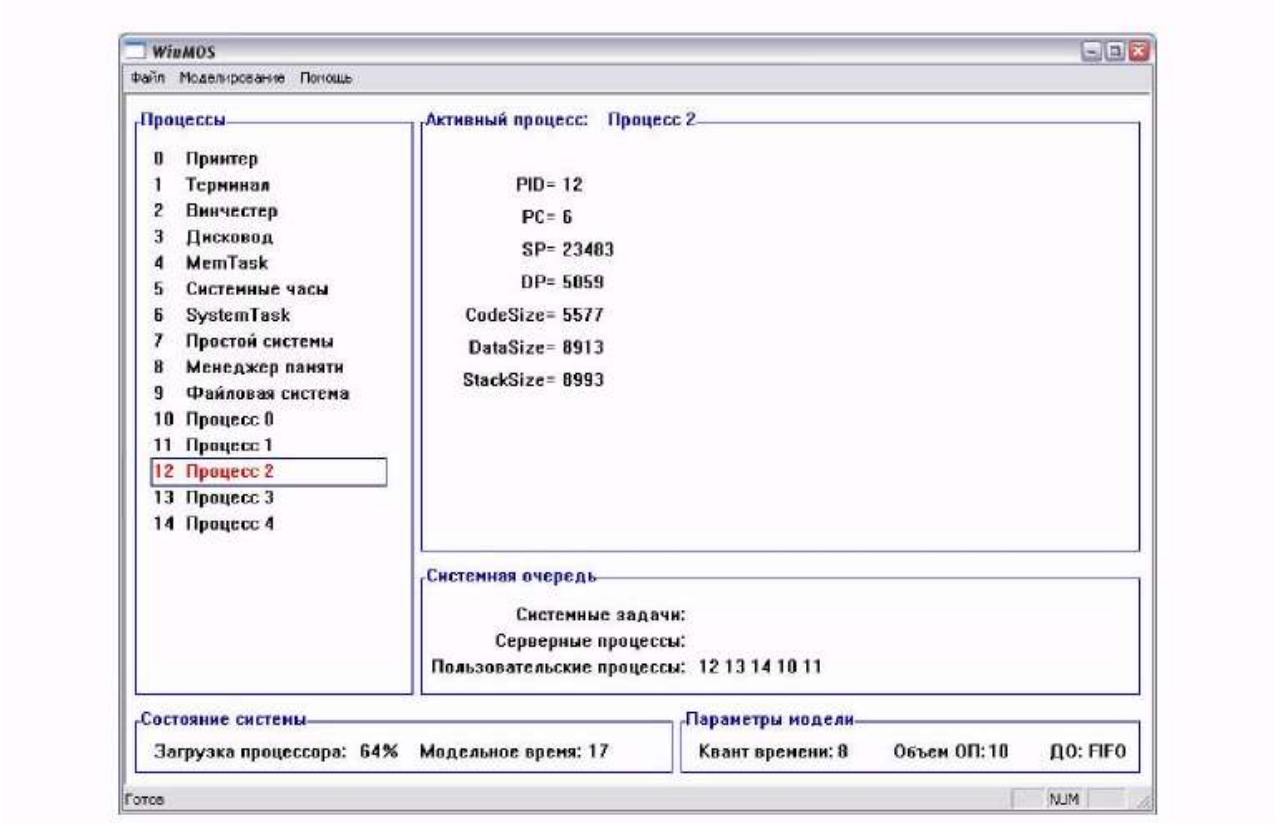


Рис. 4.6. **Режим моделювання**

Для зручності уявлення робочу область програми розподілено на вікна, обмежені синіми прямокутниками. Слід розглянути призначення і властивості кожного з них.

Вікно "Процеси". У цьому вікні відображено список процесів, що виконують у системі. Процес, що виконують у цей момент на процесорі, виділено червоним кольором. Активний процес (процес, стан якого відображено у вікні "Активний процес") виділено темно-синьою рамкою. Процеси можна активізувати шляхом натиснення мишкою на їхні імена у списку. Це дозволяє в будь-який момент бачити стан будь-якого процесу (щодо призначених для користувача процесів – дескриптор).

Вікно "Активний процес". У цьому вікні відображено статус активного процесу (виділеного синьою рамкою у вікні "Процеси"). Статус містить ідентифікатор процесу (PID), а також специфічну для процесу інформацію. Для процесів користувача – дескриптор, для менеджера пам'яті – карту пам'яті. Для файлової системи – черги до ресурсів.

Вікно "Системна черга". У цьому вікні відображено черги процесів, що очікують виділення ним кванта CPU.

Вікно "Стан системи". Це вікно містить інформацію про завантаження процесора і поточний модельний час.

Вікно "Параметри моделі". Це вікно містить інформацію про поточні параметри моделі.

Головне меню робочого вікна програми містить пункти "Файл", "Моделювання" і "Допомога". У підменю "Файл" міститься команда *Вихід*, що дозволяє вийти із системи. За допомогою підменю "Допомога" можна дізнатися інформацію про програму.

Підменю "Моделювання" містить такі пункти:

1. *Моделювання.* Вибір цього пункту приводить до появи діалогу параметрів моделювання. За його допомогою можна динамічно змінити налаштування моделювання або перезапустити модель.

2. *Результати.* Вибір цього пункту приводить до появи на екрані вікна з підсумковим графіком.

3. *Пауза.* Вибір цього пункту дозволяє припинити хід моделювання.

4. *Продовжити*. Вибір цього пункту дозволяє відновити призупинене моделювання.

Режим перегляду результатів. Зовнішній вигляд програми в режимі перегляду результатів зображено на рис. 4.7.

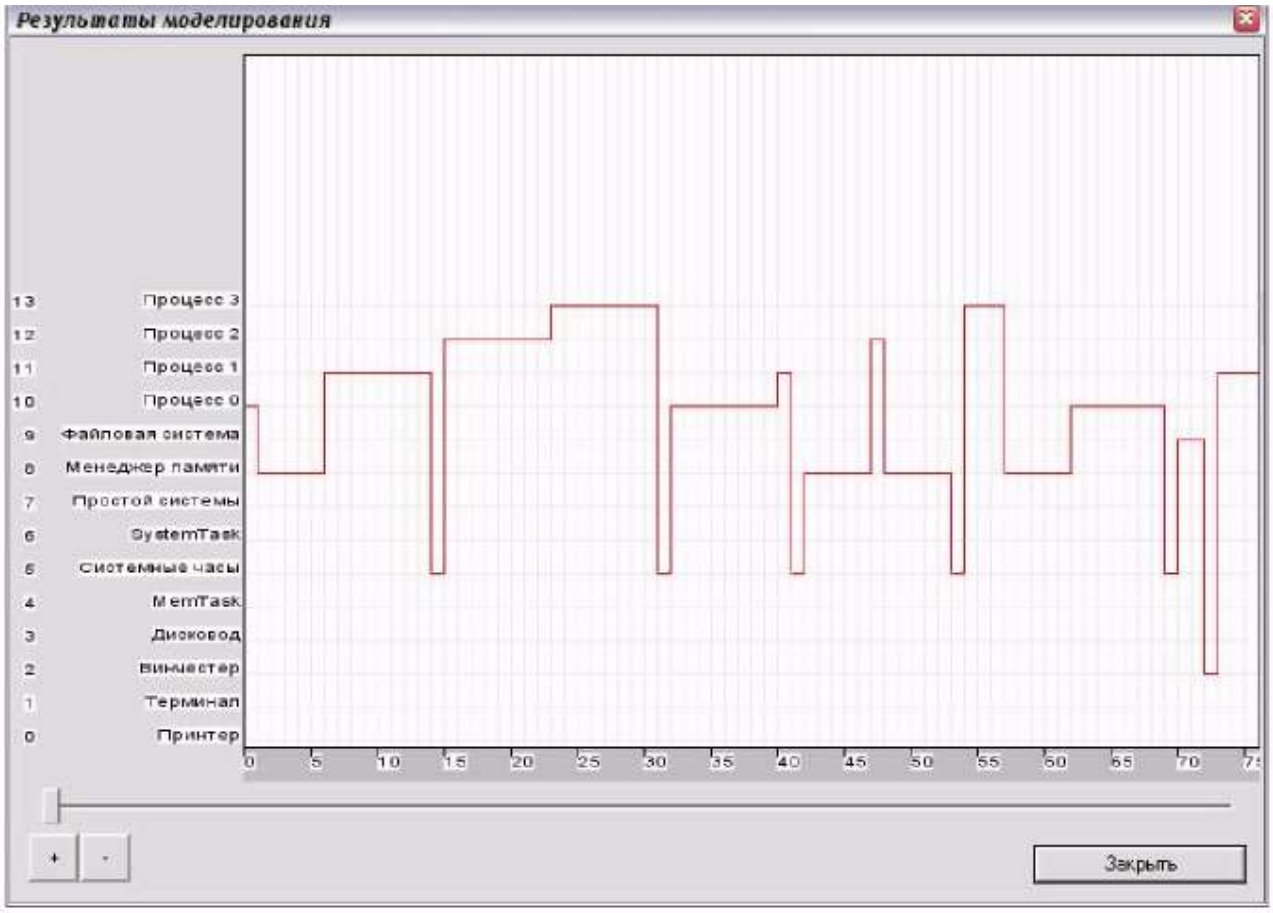


Рис. 4.7. **Тимчасова діаграма режиму перегляду результатів**

У режимі результатів програма відображає графік залежності роботи процесора від модельного часу, інакше кажучи, який процес використовував CPU в кожен конкретний момент часу.

По осі X відкладено модельний час. По осі Y – процеси, що перебувають у системі. За допомогою кнопок "+" і "−" користувач може здійснювати масштабування графіка. За допомогою бігунка здійснюють навігацію по графіку. Під час закриття діалогу програма повертається в режим моделювання.

Дослідження результуючої тимчасової діаграми. Для оцінювання точності підсумкової тимчасової діаграми формують таблицю подій, які відбуваються в системі, і порівнюють її з діаграмою. Для тестування беруть такі параметри системи: кількість процесів користувача – 3; об'єм ОП – 10 сторінок; тривалість кванта – 5; дисципліна обслуговування NUR (рис. 4.8).

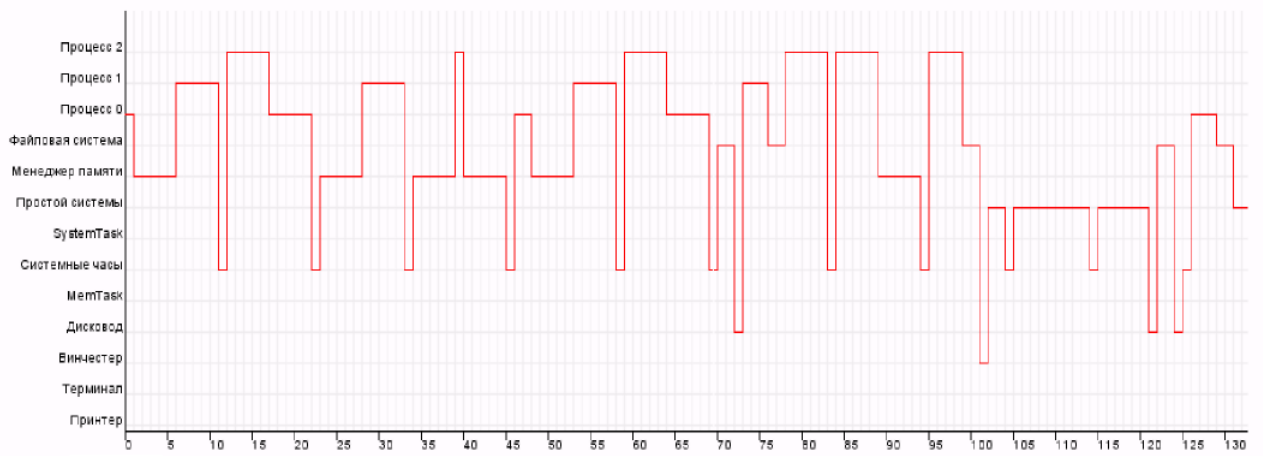


Рис. 4.8. **Підсумкова тимчасова діаграма**

Події:

1 – процес 10 викликав сторінковий перебій;

10 – у черзі з'явився процес "Системний годинник";

21 – у черзі з'явився процес "Системний годинник";

22 – процес 10 викликав сторінковий перебій;

32 – у черзі з'явився процес "Системний годинник";

33 – процес 11 викликав сторінковий перебій;

40 – процес 12 викликав сторінковий перебій;

45 – у черзі з'явився процес "Системний годинник";

48 – процес 10 викликав сторінковий перебій;

56 – у черзі з'явився процес "Системний годинник";

69 – у черзі з'явився процес "Системний годинник";

69 – процес 10 звернувся до ресурсу *Дисковід*;

76 – процес 11 звернувся до зайнятого ресурсу *Дисковід*;

80 – у черзі з'явився процес "Системний годинник";

89 – процес 12 викликав сторінковий перебій;

94 – у черзі з'явився процес "Системний годинник";

99 – процес 12 звернувся до ресурсу *Вінчестер*;

105 – у черзі з'явився процес "Системний годинник";

115 – у черзі з'явився процес "Системний годинник";

122 – процес 10 передає ресурс *Дисковід* процесу 11;

125 – у черзі з'явився процес "Системний годинник";

129 – процес 10 звернувся до зайнятого ресурсу *Вінчестер*.

Оцінювання якості моделі. Для оцінювання якості моделі необхідно з'ясувати залежність вихідних параметрів від вхідних і зіставити здобуті

результати з теоретичними положеннями. Вхідними параметрами в цій моделі є: *кількість призначених для користувача процесів, об'єм оперативної пам'яті, стратегія заміщення сторінок і тривалість кванта часу.* Вихідні параметри: *кількість сторінкових відмов, завантаження процесора* і *час завершення.* Знаходять залежності вихідних параметрів від вхідних і пояснюють здобуті результати.

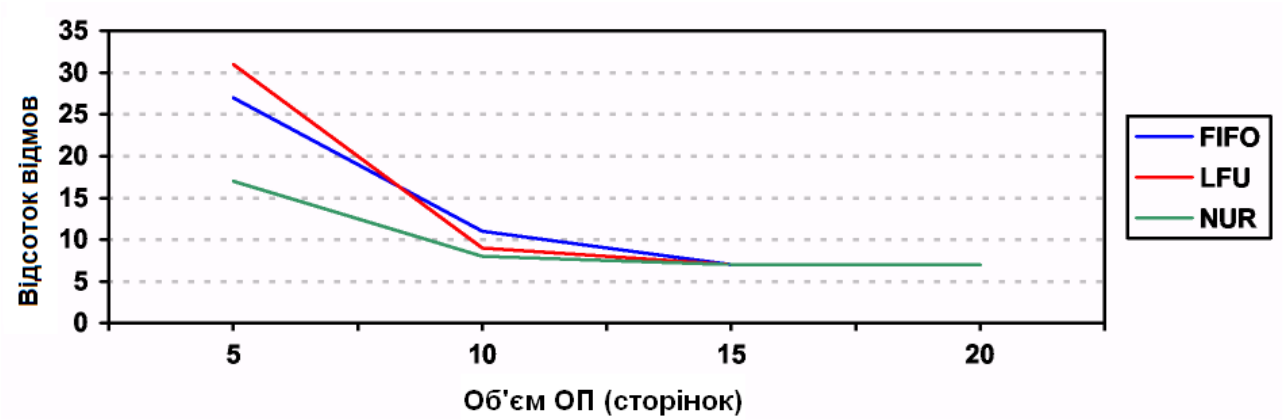
Кількість сторінкових відмов. Залежність кількості сторінкових відмов від кількості сторінок ОП та алгоритму заміщення сторінок.

Для оцінювання цієї залежності будують в одних осях графіки залежності кількості сторінкових відмов від кількості сторінок під час використання різних дисциплін обслуговування (ДО) (рис. 4.9). Для об'єктивності виконують тестування за різних значень числа призначених для користувача процесів (табл. 4.1).

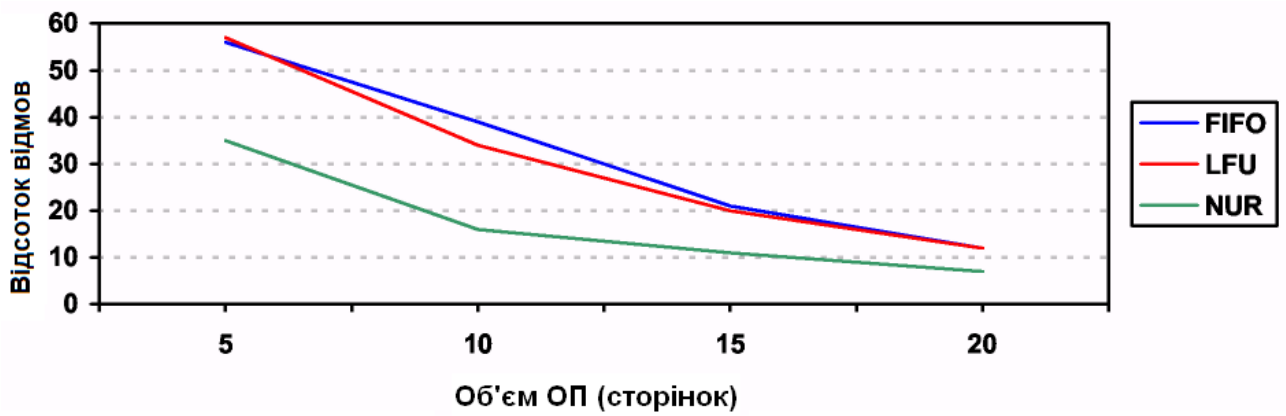
Таблиця 4.1

Результати тестування кількості сторінкових відмов

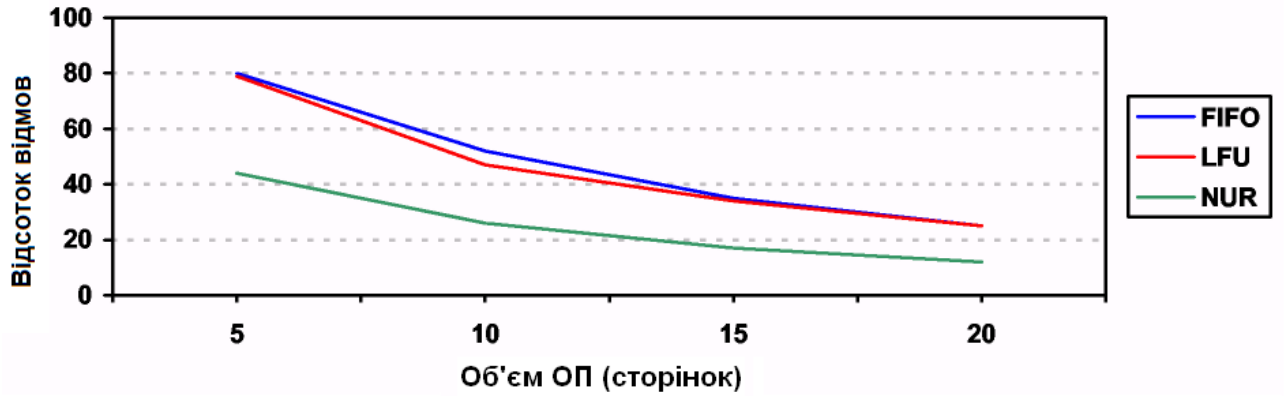
Кількість сторінок	Відсоток відмов		
	FIFO	LFU	NUR
	квант часу = 8, кількість процесів = 2		
5	27	31	17
10	11	9	8
15	7	7	7
	квант часу = 8, кількість процесів = 2		
20	7	7	7
	квант часу = 8, кількість процесів = 6		
5	56	57	35
10	39	34	16
15	21	20	11
20	12	12	7
	квант часу = 8, кількість процесів = 10		
5	80	79	44
10	52	47	26
15	35	34	17
20	25	25	12



а) параметри тестування:
квант часу = 8, кількість процесів = 2



б) параметри тестування:
квант часу = 8, кількість процесів = 6



в) параметри тестування:
квант часу = 8, кількість процесів = 10

Рис. 4.9. **Залежність відсотка сторінкових відмов від об'єму ОП**

Із графіків (див. рис. 4.9) можна зробити такі висновки:

зі збільшенням ОП кількості сторінкових відмов зменшується. Це пов'язано з тим, що збільшення ОП дозволяє розмістити більше сторінок і рідше виникає необхідність в операції вштовхування-виштовхування;

дисципліна LFU дає дещо кращі результати, ніж FIFO. Дисципліна NUR робить майже в два рази менше сторінкових відмов, ніж FIFO і LFU. Це ДО засновано на передбаченні запитів процесів і є еталонною;

на рис. 4.9 за кількістю сторінок 15 і 20 усі три ДО показують однакові результати. Це означає, що системі вдалося розмістити всі процеси ОП і операцій заміщення сторінок не відбувається. Передумови до виникнення такої ситуації дає великий об'єм ОП і малу кількість процесів;

зі збільшенням кількості процесів зростає кількість сторінкових відмов. Це результат дефіциту ресурсу пам'ять для великої кількості процесів.

Залежність кількості сторінкових відмов від кванта процесорного часу

Передбачають, що кількість сторінкових відмов не залежить від кванта часу, проте цю гіпотезу необхідно перевірити.

Було оцінено залежність кількості сторінкових відмов від параметра моделювання квантів часу під час установлення системи: 5 призначених для користувача процесів (табл. 4.2).

Таблиця 4.2

Результати тестування залежності кількості відмов від кванта часу
(об'єм ОП = 10, ДО – LFU)

Квант часу	Кількість відмов
1	27
5	28
8	28
10	25
15	25
20	25

Виявилося незначне зменшення кількості сторінкових відмов за великих значень кванта часу (рис. 4.10). Мабуть, це пов'язано зі збільшенням часу безперервного перебування кожного процесу на процесорі, що супроводжено зверненнями до одних і тих самих сторінок пам'яті та як наслідок – деяким зменшенням хаотичності запитів до сторінок пам'яті, що призводить до зменшення кількості сторінкових відмов.

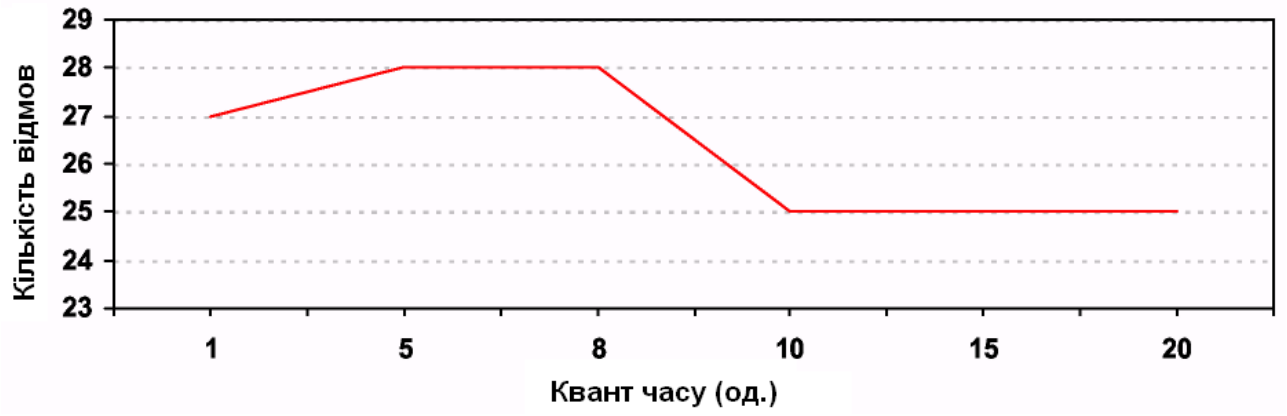


Рис. 4.10. **Залежність кількості відмов від кванта часу**

Завантаження процесора – це параметр, що змінюється в часі. Тому оцінювали залежність Z = CPU(TM). Це дуже важливий показник, який характеризує якість роботи системи загалом.

Оцінювання залежності завантаження процесора від алгоритму заміщення сторінок. Для оцінювання цієї залежності будують в одних осях графіки залежності завантаження процесора від часу під час використання різних ДО, на яких передбачають зафіксувати такі три інтервали:

1. *Інтервал зростання завантаження.* У цей час призначені для користувача процеси реєструються у системі та починають формувати запити. Процесор поступово завантажується до можливого максимуму.

2. *Інтервал максимуму.* Протягом цього інтервалу відбувається основна робота. Процесор надається процесам по черзі. Завантаження процесора на цьому етапі максимальне.

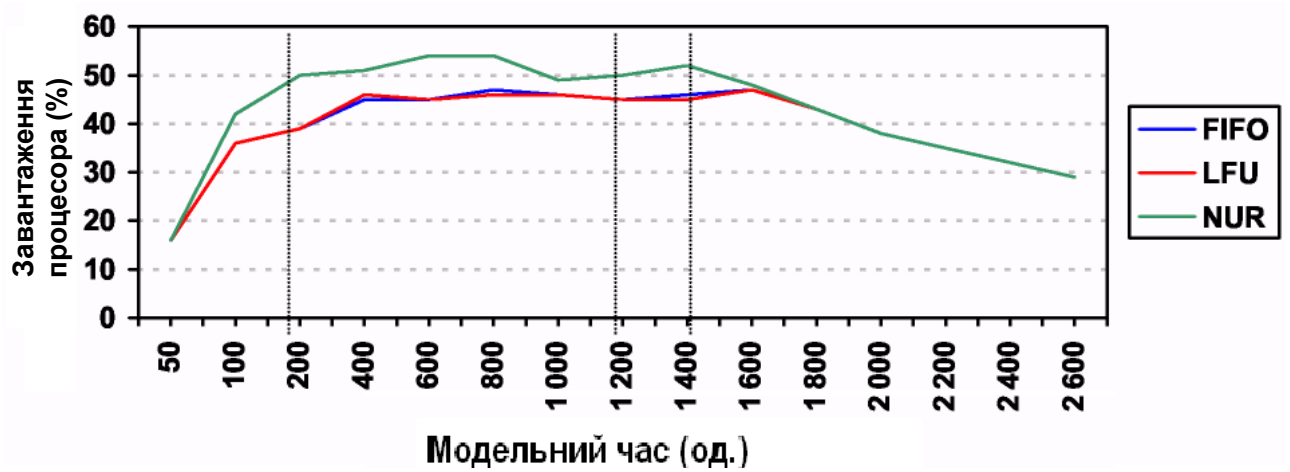
3. *Інтервал убування.* Спадання відбувається, запити користувачевих процесів уже оброблено, і процеси не потребують роботи на процесорі. Чим раніше настає цей етап, тим краще працює система.

Для об'єктивності виконують тестування за різних значень числа призначених для користувача процесів (табл. 4.3, рис. 4.11).

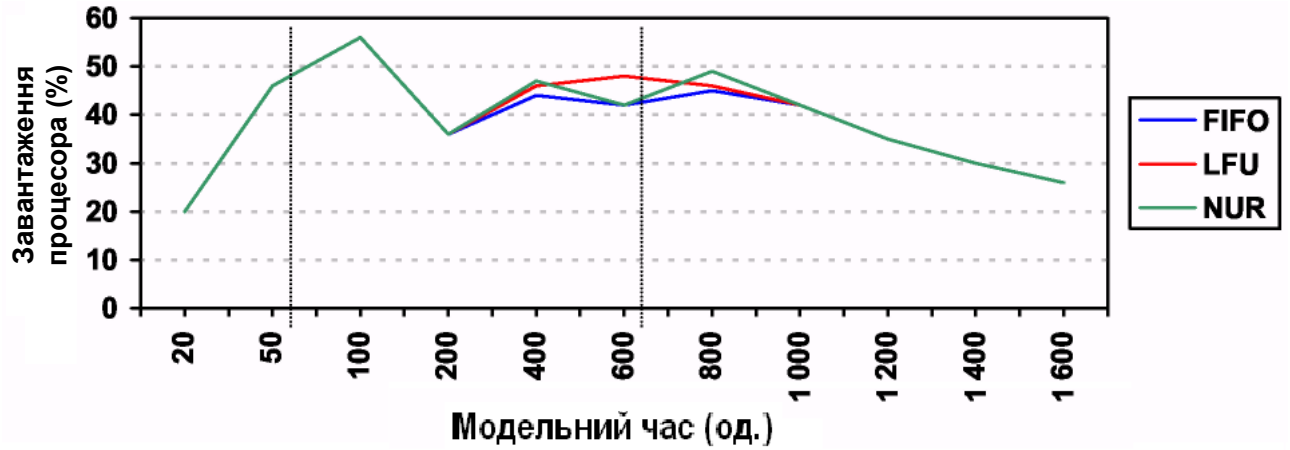
Таблиця 4.3

Результати тестування завантаження процесора в часі

ТМ	Завантаження CPU		
	FIFO	LFU	NUR
	квант часу = 8, кількість процесів = 8, об'єм ОП = 10		
50	16	16	16
100	36	36	42
200	39	39	50
400	45	46	51
600	45	45	54
800	47	46	54
1 000	46	46	49
1 200	45	45	50
1 400	46	45	52
1 600	47	47	48
1 800	43	43	43
2 000	38	38	38
2 200	35	35	35
2 400	32	32	32
2 600	29	29	28
	квант часу = 8, кількість процесів = 8, об'єм ОП = 10		
20	20	20	20
50	46	46	46
100	56	56	56
200	36	36	36
400	44	46	47
600	42	48	42
800	45	46	49
1 000	42	42	42
1 200	35	35	35
1 400	30	30	30
1 600	26	26	26



а) параметри тестування:
квант часу = 8, кількість процесів = 8, об'єм ОП = 10



б) параметри тестування:
квант часу = 8, кількість процесів = 3, об'єм ОП = 10

Рис. 4.11. **Залежність завантаження CPU від модельного часу**

Із графіків залежностей (див. рис. 4.11) можна зробити такі висновки:

на графіках достатньо точно простежують інтервали зростання, максимуму та спадання;

для восьми процесів (рис. 4.11а) дисципліна NUR дає більше завантаження процесора, ніж LFU і FIFO. Це пов'язано з меншою кількістю сторінкових відмов, що дають ДО = NUR. Для трьох призначених для користувача процесів (рис. 4.11б) різні ДО показують схожі результати. Це пояснено тим, що системі вдалося повністю, або майже повністю, розмістити процеси ОП (у зв'язку з малою їхньою кількістю), і кількість сторінкових відмов мале, незалежно від алгоритму заміни сторінок;

для восьми процесів інтервал спадання для дисципліни NUR почався раніше. Це означає, що системі вдалося швидше задовольнити запити призначених для користувача процесів, і ще раз підтверджує високу ефективність дисципліни NUR;

дисципліни LFU і FIFO показали схожі результати. Отже, деяка перевага LFU перед FIFO не позначається значним чином на завантаженні процесора.

Оцінювання залежності завантаження процесора від величини кванта часу

Для оцінювання цієї залежності будують в одних осях графіки залежності завантаження процесора від часу за різних значень кванта часу. Конфігурація системи під час тестування: 5 призначених для користувача процесів, ДО = FIFO, об'єм пам'яті 5 сторінок (табл. 4.4).

Таблиця 4.4

Результати тестування залежності завантаження процесора від величини кванта часу

ТМ	Завантаження CPU, %			
	Квант = 1	Квант = 7	Квант = 14	Квант = 20
30	16	16	16	16
60	43	41	41	41
100	42	41	40	40
200	34	35	36	36
400	42	47	49	42
600	44	47	52	46
800	46	49	54	49
1 000	46	46	49	46
1 200	46	49	51	48
1 400	47	48	48	48
1 600	46	47	47	46
1 800	42	42	42	42
2 000	38	38	38	38
Середнє завантаження	40,92	42,00	43,31	41,38

На графіку (рис. 4.12) видно області з наростання, максимуму і спадання завантаження процесора, проте залежність завантаження процесора від кванта часу не очевидна.

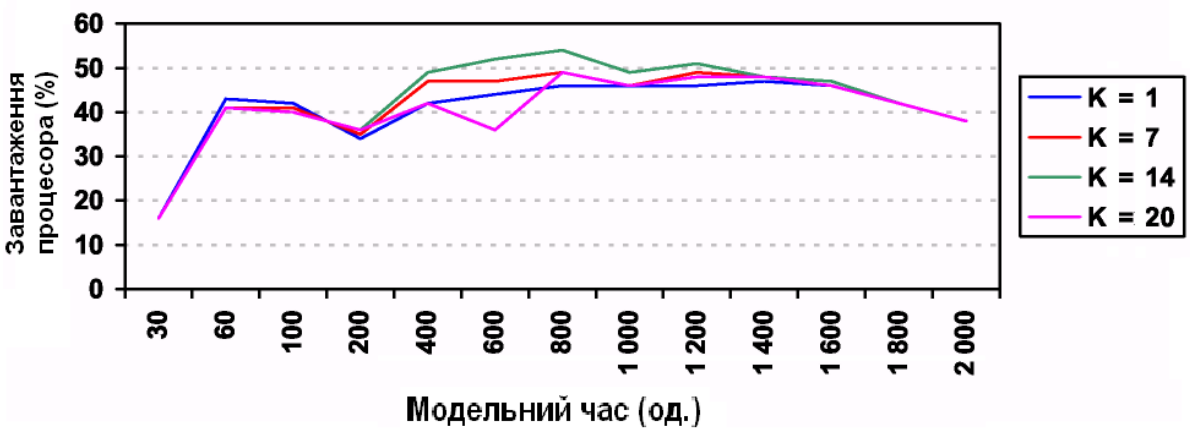


Рис. 4.12. **Залежність завантаження CPU від модельного часу**

Для наочності обчислюють середнє завантаження процесора за час роботи системи для кожного значення кванта часу (див. табл. 4.4) і будують графік залежності середнього завантаження процесора від тривалості кванта (рис. 4.13).

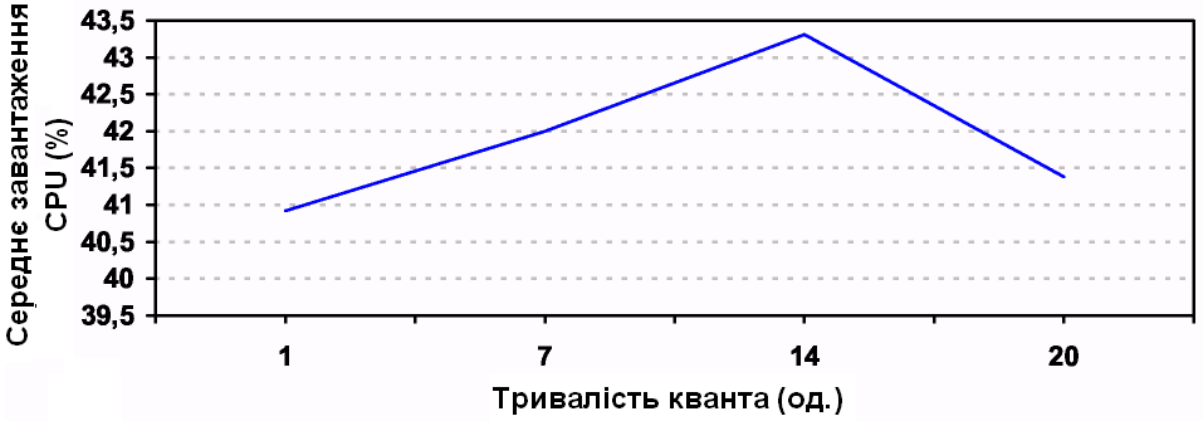


Рис. 4.13. **Залежність завантаження CPU від тривалості кванта часу**

Із графіка (див. рис. 4.13) можна зробити такі висновки:

за невеликої тривалості кванта з її зростанням збільшується і завантаження процесора. Це може бути пояснено тим, що чим триваліший час безперервного використання процесом CPU, тим більша кількість запитів може сформувати процес, і в результаті з вищою ймовірністю не буде чекати в системній черзі чергового кванта процесорного часу, а займе який-небудь ресурс. Крім того, зі збільшенням кванта часу система рідше буде виконувати власні завдання, наприклад, системний годинник;

за великої тривалості кванта завантаження процесора спадає. Це пов'язано з тим, що система менш оперативно відповідає на запити процесів.

Загальне завдання для виконання

Розробіть модель функціонування ОС під час оброблення системних і призначених для користувача процесів і проведіть дослідження її параметрів за заданим варіантом (табл. 4.5).

Крім того, **обов'язковими параметрами** для моделі є (рис. 4.14):

1. Використання випадкових запитів.
2. Збільшення вірогідності тупика.
3. Ручне трасування.

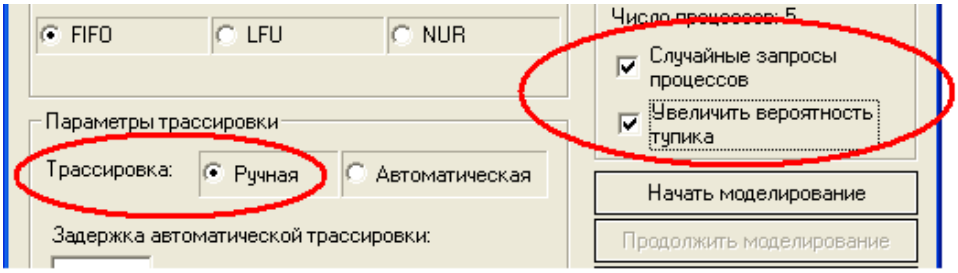


Рис. 4.14. **Обов'язкові параметри налаштування моделювання**

Таблиця 4.5

Індивідуальні завдання для виконання

Варіанти	Розмір кванта	Об'єм оперативної пам'яті	Кількість процесів користувача	Стратегія заміщення (K)	Тривалість моделювання
1	2	3	4	5	6
1	1	18	2	FIFO, LFU	340 + $№_{журн.}$*
2	2	19	3	LFU, NUR	320 + $№_{журн.}$
3	3	20	4	NUR, FIFO	300 + $№_{журн.}$
4	4	7	5	FIFO, LFU	280 + $№_{журн.}$
5	5	8	6	LFU, NUR	260 + $№_{журн.}$
6	6	9	7	NUR, FIFO	240 + $№_{журн.}$
7	7	10	8	FIFO, LFU	220 + $№_{журн.}$
8	8	11	9	LFU, NUR	200 + $№_{журн.}$
9	9	12	10	NUR, FIFO	180 + $№_{журн.}$
10	10	13	2	FIFO, LFU	360 + $№_{журн.}$
11	11	14	3	LFU, NUR	340 + $№_{журн.}$
12	12	15	4	NUR, FIFO	320 + $№_{журн.}$

Закінчення табл. 4.5

1	2	3	4	5	6
13	13	16	5	FIFO, LFU	300 + №$_{журн.}$
14	14	1	6	LFU, NUR	280 + №$_{журн.}$
15	15	2	7	NUR, FIFO	260 + №$_{журн.}$
16	16	3	8	FIFO, LFU	240 + №$_{журн.}$
17	17	4	9	LFU, NUR	220 + №$_{журн.}$
18	18	5	10	NUR, FIFO	200 + №$_{журн.}$
19	19	6	2	FIFO, LFU	360 + №$_{журн.}$
20	20	7	3	LFU, NUR	340 + №$_{журн.}$
21	7	8	4	NUR, FIFO	320 + №$_{журн.}$
22	8	9	5	FIFO, LFU	300 + №$_{журн.}$
23	9	10	6	LFU, NUR	280 + №$_{журн.}$
24	10	11	7	NUR, FIFO	260 + №$_{журн.}$
25	11	12	8	FIFO, LFU	240 + №$_{журн.}$
26	12	13	9	LFU, NUR	220 + №$_{журн.}$
27	13	14	10	NUR, FIFO	200 + №$_{журн.}$
28	14	15	2	FIFO, LFU	380 + №$_{журн.}$
29	15	16	3	LFU, NUR	340 + №$_{журн.}$
30	16	17	4	NUR, FIFO	300 + №$_{журн.}$

* № журн. – номер студента в журналі групи.

Імена призначених для користувача процесів задають у такому форматі: XX_YY_ZZ_#, де XX – перші дві букви прізвища студента; YY – ініціали студента; ZZ – дві цифри місяця народження студента; # – порядковий номер самого процесу. До того ж **усі імена процесу задають великими буквами англійського алфавіту.** Задані за варіантом параметри та початковий стан моделі перед стартом (початком моделювання) передають у звіті. У процесі моделювання необхідно отримати та подавати у звіті (у вигляді графіків, таблиць) таку інформацію:

1. Кількість викликів системного годинника, шт.

2. Кількість викликів завдань, обслуговуючих ресурси (*Принтер, Термінал, Вінчестер, Дисковід*), – за кожним ресурсом окремо.

3. Кількість викликів System Task, шт.

4. Кількість простоїв і їхня загальна тривалість у квантах.

5. Кількість сторінкових відмов.

6. Кількість операцій P(S) і V(S), семафор якого пристрою спрацював, хто був власником ресурсу, який процес очікував і який процес звільнив ресурс (табл. 4.6).

Таблиця 4.6

Динаміка використання ресурсів при моделюванні

№ п/п	Квант часу	P(S)	V(S)	Семафор	Власник, PID	Черга процесів			Звільнювальний процес, PID
Початковий стан									
				Вінчестер	−1				
				Дисковід	−1				
				Термінал	−1				
				Принтер	−1				
Моделювання									
1									

7. У діапазонах часу квантів 0 … 10, 50 ... 60, 100 … 110, 150 … 160, 200 … 210, 250 … 260, 300 … 310, 350 … 360 тощо необхідно зареєструвати всі параметри призначених для користувача процесів, активних у цей момент, як показано на рис. 4.15 (табл. 4.7).

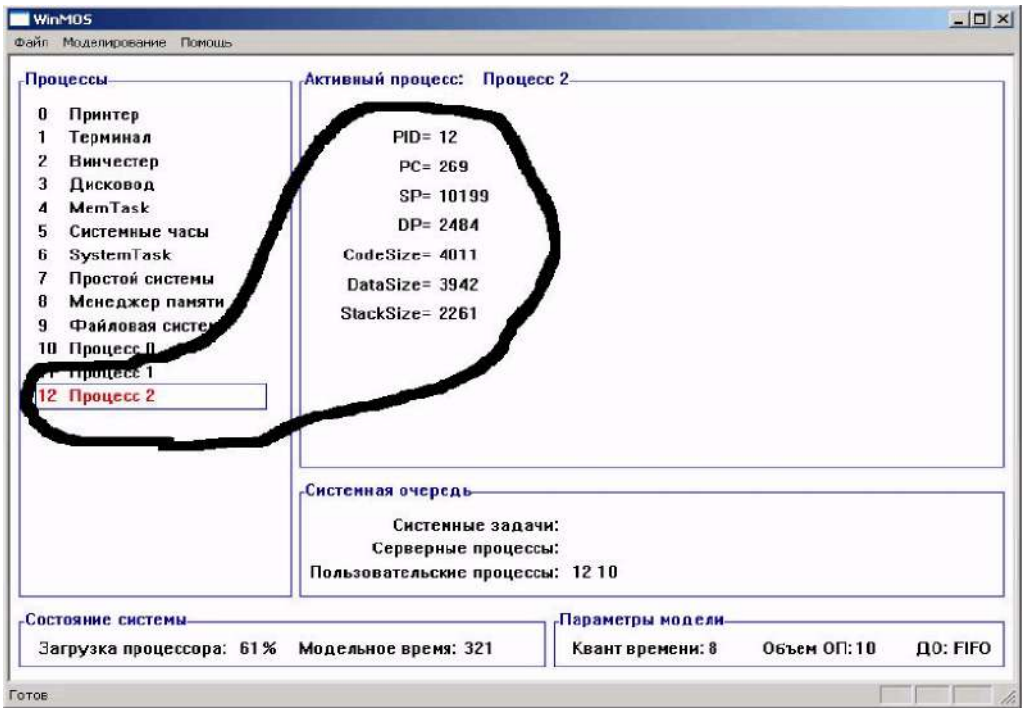


Рис. 4.15. **Параметри активного процесу, призначеного для користувача**

Таблиця 4.7

Динаміка виконання процесів, призначених для користувача

№ п/п	Квант часу	PID	PC	SP	DP	CodeSize	DataSize	StackSize
1	1	11	269	199	243	4 011	3 942	2 261
2	7	12	290	200	243	4 016	3 972	5 261

8. Результати моделювання (графік залежності роботи процесора від модельного часу із програми з його описом без масштабування склеєний по всій довжині тривалості експерименту) необхідно подати у звіті.

9. Залежності завантаження процесора від алгоритму заміщення сторінок (у вигляді таблиць і відповідних графіків), як показано у прикладах на рис. 4.10 і табл. 4.3, подати у звіті. Дані обробляти для кількості квантів часу **X + 50**, де X – задана за варіантом кількість квантів часу; об'єм ОП фіксований. Надати висновки за цими залежностями.

10. Залежність кількості сторінкових відмов від кількості сторінок ОП і алгоритму заміщення сторінок (у вигляді таблиць і відповідних графіків), як показано у прикладах на рис. 4.8 і табл. 4.1, подати у звіті. Дані обробляти для кількості квантів часу **X + 50**, де X – задана за варіантом кількість квантів часу; і числа призначених для користувача процесів **M + 2**, де M – задана за варіантом кількість призначених для користувача процесів. Надати висновки за цією залежністю.

11. Залежність кількості сторінкових відмов від квантів процесорного часу (у вигляді таблиці та відповідного графіка), як показано у прикладах на рис. 4.9 і табл. 4.2, подати у звіті. Надати висновки за цією залежністю.

12. Залежність завантаження процесора від величини кванта часу (у вигляді таблиць і відповідних графіків), як показано у прикладах на рис. 4.11 і табл. 4.4, подати у звіті. Дані, при цьому, обробляти для кількості квантів часу із кроком 50. Надати висновки за цими залежностями.

13. Графік залежності середнього завантаження процесора від тривалості кванта на основі прикладу на рис. 4.12 і даних, отриманих у п. 12, подати у звіті. Надати висновки за цією залежністю.

Зміст звіту

Звіт має містити:

1. Назву, дату виконання, тему, мету, опис завдання, матеріальне забезпечення, хід виконання та результати лабораторної роботи.

2. Загальні висновки за пп. 1 – 13 і загальні висновки по роботі.

Контрольні запитання

1. Яким чином взаємодіють процеси під час моделювання?

2. Як взаємодіють резидентна і нерезидентна частина таблиці управління процесами?

3. Яким чином здійснюють синхронізацію процесів?

4. Поясніть особливості планування процесів.

5. Поясніть особливості алгоритмів заміщення сторінок та вимоги до них.

6. Яку роль у лабораторній роботі відведено семафору? Назвіть його можливі значення.

7. Скільки рівнів захисту простору пам'яті та введення/виведення підтримує захищений режим процесора?

8. Скільки функцій експортується із процесу підсистеми оточення?

9. Де зберігають відомості про процеси?

10. Що таке "інкапсуляція об'єктів"?

11. Що таке "метод об'єкта"?

12. Укажіть номер пріоритету програмного переривання для потоку обнулення сторінок.

13. Укажіть символ, який використовують для позначення кореня ієрархічного дерева.

14. Чим визначено можливість операційної системи виконувати додатки, розроблені для інших операційних систем?

15. Чому має дорівнювати значення лічильника кількості користувачів об'єкта, щоб він був знищений?

16. Що визначає взаємовідносини одних модулів ОС з іншими?

17. Що повертається диспетчером об'єктів у разі створення об'єкта?

18. Що входить до контексту потоку?

19. Що входить у поняття архітектури ОС?

20. Що є причиною переходу процесу у стан блокування?

21. Що є причиною перемикання режимів роботи ОС?

Лабораторна робота 5
Дослідження властивостей процесів і потоків

Мета роботи: ознайомлення з концепцією мультипрограмування і багатопотоковості в сучасних операційних системах, набуття практичних навичок у тестуванні, аналізі зареєстрованих у них процесів і потоків, виділенні специфічних властивостей і характеристик процесів за рахунок використання різних програмних продуктів, здійсненні аналітичного порівняння цих характеристик.

Рекомендації з підготовки до виконання лабораторної роботи

Найважливішою частиною операційної системи, що безпосередньо впливає на функціонування обчислювальної машини, є підсистема управління процесами. Підсистема управління процесами планує виконання процесів, тобто розподіляє процесорний час між декількома процесами, що одночасно відбуваються у системі, а також займається створенням і видаленням процесів, забезпечує процеси необхідними системними ресурсами, підтримує взаємодію між процесами.

Таким чином, необхідно вивчити порядок створення та видалення процесів (потоків) в операційній системі. Під час підготовки до лабораторної роботи необхідно ознайомиться з теоретичним описом процесів, потоків і волокон, принципу багатозадачності й особливостями їхньої апаратної реалізації.

Додаткову інформацію під час підготовки до роботи можна отримати в [5; 6].

Теоретичні відомості

В операційній системі Windows підтримують багатопотокові процеси. **Процес** (process) – це об'єкт, якому належать ресурси додатка. **Потік** (thread) – це суть усередині процесу, яку ядро Windows спрямовує на виконання. Без нього програма процесу не може виконуватися. Потік розподіляє разом із процесом загальний адресний простір, код і глобальні дані. У кожного потоку є власні регістри, стек і механізми введення, зокрема і черга прихованих повідомлень.

Багатозадачність (multitasking) – це можливість операційної системи виконувати декілька програм одночасно. Основою цього принципу є використання операційною системою апаратного таймера для виділення відрізків часу для будь-якого з одночасно виконуваних процесів. Якщо ці відрізки часу досить маленькі, і процесор не переобтяжений дуже великою кількістю програм, то користувачеві здасться, що всі ці програми виконуються паралельно.

Багатопотоковість – це можливість програми самою бути багатозадачною. Програму може бути розподілено на окремі потоки виконання, що виконують паралельно.

Для дослідження процесів і потоків є дві основні групи програмних засобів: убудовані в операційну систему та сторонніх виробників.

Основною вбудованою програмою є *Диспетчер завдань Windows* (Task Manager Windows, %SYSTEMROOT%\system32\taskmgr.exe).

До програм сторонніх виробників, розгляду яких присвячено лабораторну роботу, належать:

1. *Aida* (Worldwide Sysinfo Tool, by Tamas Miklos).
2. *Process Explorer & Process Monitor* (by Mark Russinovich).
3. *System Info for Windows* (by Gabriel Topala).
4. *Task Info* (Iarsn System Software).
5. *Microsoft Spy++* (Microsoft Corporation) та ін.

Саме дослідженню за допомогою вказаних програм присвячено цю лабораторну роботу.

Диспетчер завдань Windows. У *Диспетчерові завдань* відображаються відомості про програми і процеси, що виконують на комп'ютері. Крім того, там можна проглянути найбільш часто використовувані показники швидкодії процесів.

Диспетчер завдань слугує для відображення ключових показників швидкодії комп'ютера. Для виконуваних програм можна проглянути їхній стан і завершити програми, що перестали відповідати на запити. Є можливість перегляду активності процесів, що виконують, із використанням до 15 параметрів, а також графіків і відомостей про використання центрального процесора (ЦП) і пам'яті.

Крім того, якщо є підключення до мережі, можна проглядати стан мережі та параметри її роботи. Якщо до комп'ютера підключилися декілька користувачів, можна побачити їхні імена, які завдання вони виконують, а також відправити їм повідомлення (рис. 5.1).

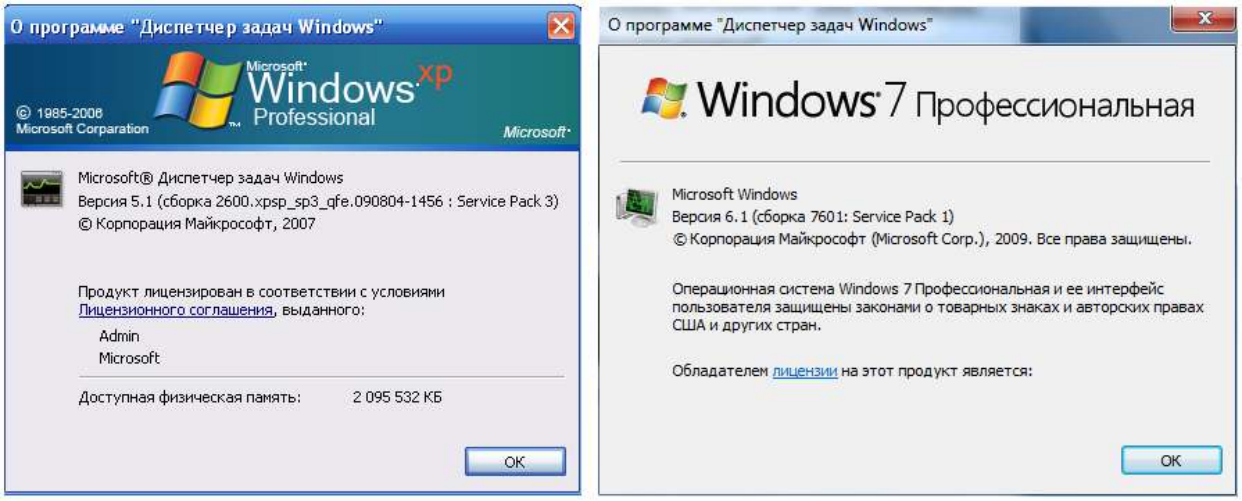


Рис. 5.1. **Діалогове вікно *Диспетчера завдань* Windows**

Для управління процесами *Диспетчер завдань* Windows містить вкладку "Процеси". На ній є можливість відстежувати виконання процесів за допомогою таких лічильників, що відображають у заголовках стовпців:

1. Ім'я образу – це ім'я процесу в *Диспетчері завдань.*

2. Ідентифікатор процесу (PID) – це числовий ідентифікатор, використовуваний для позначення процесу під час виконання.

3. Базовий пріоритет – це ранг, що визначає порядок, у якому потоки процесу обробляються процесором. Зміну пріоритету виконують за допомогою пункту "Пріоритет" контекстного меню. Водночас можливі такі значення (табл. 5.1):

Таблиця 5.1

Класи пріоритетів процесу

№ п/п	Класи пріоритетів	Значення	Опис
1	2	3	4
1	Реального часу	24	Процес із таким пріоритетом витісняє навіть дії операційної системи. Цей клас слід застосовувати тільки в разі абсолютної необхідності
2	Високий	13	Процес із високим пріоритетом має вплив на можливість виконання інших процесів

Закінчення табл. 5.1

1	2	3	4
3	Вищий за середній	10	Цей прапор позначає процес, пріоритет якого перевищує середній клас, проте нижчий за високий клас
4	Середній	8	Основний пріоритет процесу (заданий за замовчуванням)
5	Нижчий за середній	6	Цей прапор позначає процес, пріоритет якого перевищує низький клас, проте нижчий за середній клас
6	Низький	4	Процес із найнижчим пріоритетом, виконується в період простою системи

4. Час ЦП – це загальний процесорний час (у секундах), використаний процесом із моменту свого запуску.

5. Завантаження ЦП – це відсоткова частка часу, протягом якого процес використовував ЦП із моменту останнього оновлення.

6. Об'єм віртуальної пам'яті – це розмір адресного простору, переданий процесу.

7. Пам'ять максимум – це об'єм фізичної пам'яті, використовуваної процесом із моменту його запуску.

8. Пам'ять використання – це поточний набір сторінок пам'яті, зайнятих процесом (у КБ). Поточний набір – це кількість сторінок, резидентних зараз у пам'яті.

9. Пам'ять зміна – це об'єм пам'яті (у КБ), використаної з моменту останнього оновлення процесом.

10. Об'єкт USER – це об'єкт *Диспетчера вікон*, що містить вікна, меню, курсори, значки, обробники, поєднання клавіш та інші внутрішні об'єкти.

11. Об'єкт GDI – це об'єкт бібліотеки GDI-інтерфейсів програмування для графічних пристроїв.

12. Лічильник потоків – це кількість потоків, що виконуються процесом.

13. Лічильник дескрипторів – це кількість дескрипторів об'єктів таблиці об'єктів процесу в *Диспетчері завдань.*

14. Вивантажуваний пул (куча) – це віртуальна пам'ять, виділена системою процесу для розміщення програмного коду та даних, які можуть бути вивантажені з оперативної пам'яті. Підкачування сторінок – це переміщення рідко використовуваних частин коду програми з оперативної пам'яті на системний носій. Зазвичай використовують жорсткий диск.

15. Невивантажуваний пул – це пам'ять, зайнята операційною системою, ніколи не вивантажувана на диск та ін.

Єдиний із лічильників, яким може управляти користувач – це базовий пріоритет процесу.

Aida. Безкоштовна професійна програма, яку випускають у двох різних типах версій: Aida32 і Aida64.

Aida32 – це професійний інструмент для діагностики обладнання й аналізу системної конфігурації. Аналізує комп'ютер і видає докладну інформацію як про його апаратну частину (процесор, материнську плату, монітор і відеопідсистему повністю, диски тощо), так і про програмну начинку (операційну систему, драйвери, усі встановлені й окремо автозавантажувані програми, запущені процеси, ліцензії тощо (рис. 5.2)).

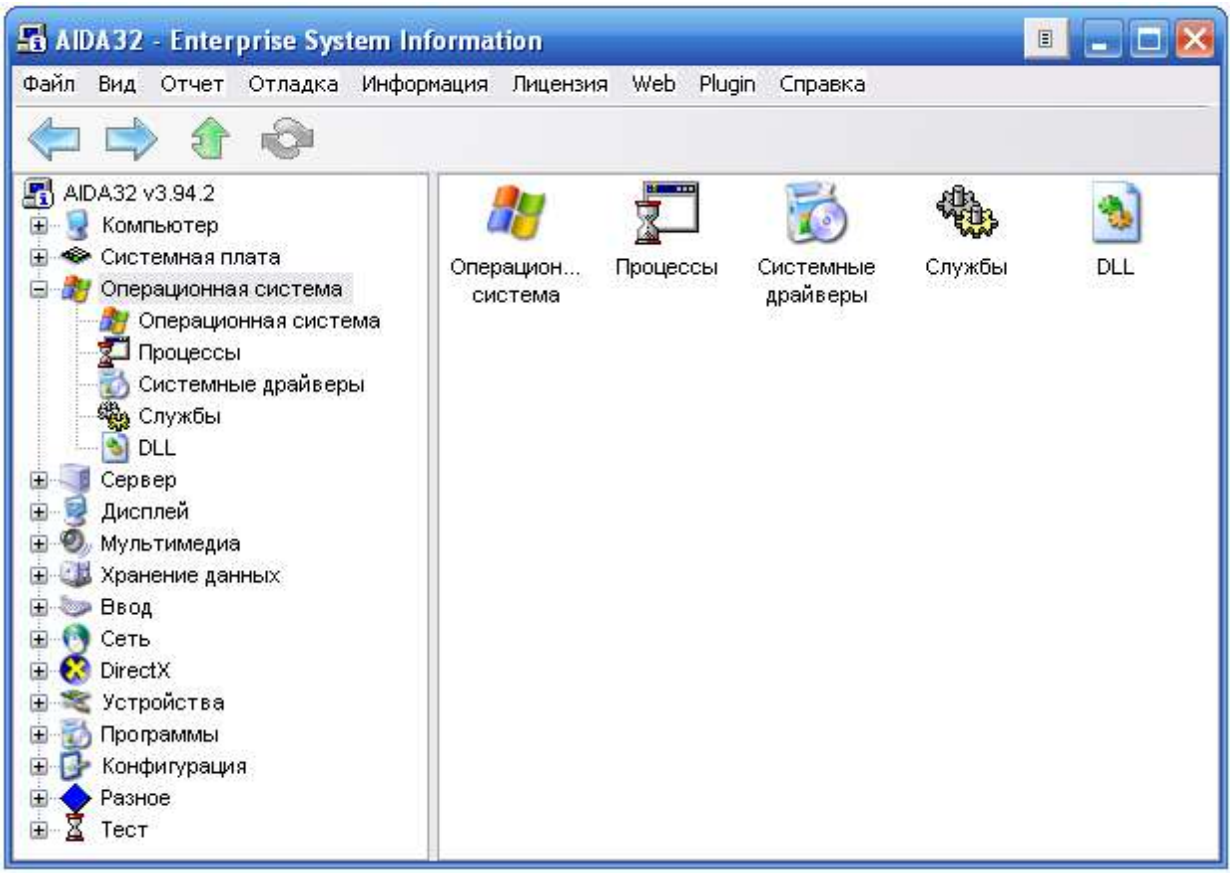


Рис. 5.2. **Головне вікно програми Aida32**

Aida32 отримує дані про пристрої на низькому рівні (а не тільки за системним реєстром Windows, як більшість Win32-інформерів), використовуючи власну базу даних (близько 21 000 пристроїв). Aida32 дозволяє

збирати інформацію з видалених комп'ютерів у мережі TCP/IP. Зокрема Aida32 показує такі характеристики:

1) апаратна конфігурація: центральний процесор, материнська плата, монітор, відеоплата, установлені жорсткі диски, BIOS тощо;

2) програмна конфігурація: інформація і стан операційної системи, установлені драйвери та програмне забезпечення, об'єкти автозапуску, поточні відкриті процеси та служби, оновлення тощо.

Так само програма уміє тестувати продуктивність комп'ютера і звіряти її з еталонними результатами для відображення загальної картини швидкості цього комп'ютера.

Aida64 (колишній EVEREST) – це утиліта, що є могутнім засобом для ідентифікації й тестування практично будь-яких компонентів персонального комп'ютера під управлінням операційних систем сім'ї Windows (рис.5.3).

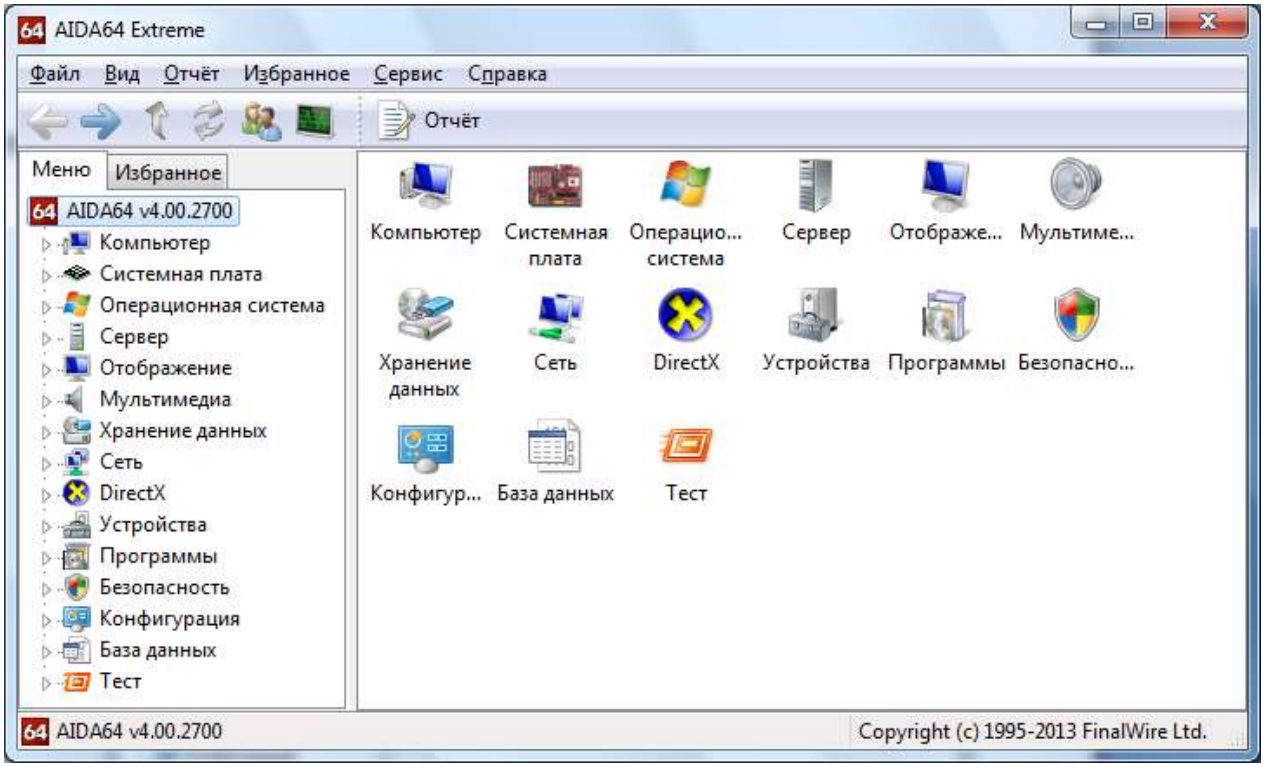


Рис. 5.3. **Головне вікно програми Aida64**

У процесі оптимізації та точного налаштування програма забезпечує отримання необхідної системної інформації, надає просунуті можливості моніторингу й діагностики апаратного забезпечення для оцінювання

ефекту, досягнутого застосуванням тих або тих налаштувань. Тести продуктивності центрального процесора, модуля обчислень із плаваючою точкою, а також пам'яті допомагають визначати реальну продуктивність системи та порівняти її з раніше здобутими результатами або з іншими комп'ютерами.

Process Explorer – це компактна, але могутня програма зі зручним інтерфейсом для моніторингу процесів, що відбуваються в системі, у режимі реального часу. Видає докладну інформацію про всі запущені процеси, включаючи власника, використання пам'яті, задіяні бібліотеки тощо (рис. 5.4).

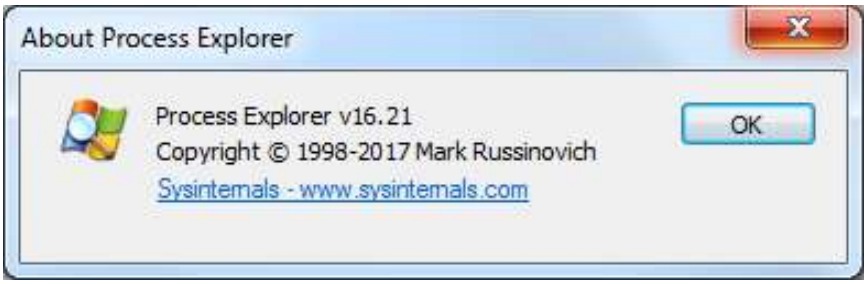


Рис. 5.4. **Діалогове вікно про програму Process Explorer**

Process Explorer володіє могутньою системою пошуку, що дозволяє шукати процеси, що відкривають специфічний дескриптор або завантажують певну DLL.

За допомогою Process Explorer можна зберегти в текстовому файлі список усіх процесів з описами та розміром зайнятої кожним із них пам'яті, запустити будь-який додаток, вимкнути, перезавантажити або заблокувати комп'ютер, знайти використовувані бібліотеки, увімкнути підсвічування кольором певних процесів тощо. Усе це допомагає контролювати процеси, що працюють та отримувати вичерпну інформацію про використання системних ресурсів.

Основні можливості Process Explorer:

ієрархічне відображення процесів;

можливість ідентифікації системних процесів (наприклад, чи є процес svchost.exe системним або "лівим");

відображає іконку і компанію виробника кожного процесу;

змінний діапазон вимірювань завантаження CPU і графічні індикатори;

можливість "заморозити" будь-який процес;

можливість управління (запуск, пауза, зупинка) потоками процесу;

можливість закриття дерева процесів;

можливість вивести вікно, що належить тому або тому процесу зверх останніх;

можливість у реальному режимі часу міняти пріоритет і те, на якому ядрі процесора будуть виконувати той або той процес;

можливість перевірки сертифіката файла процесу;

можливість замінювати системний *Диспетчер завдань* за тими самими гарячими клавішами;

для кожного об'єкта, який має access control list (ACL), відображається вкладка "Безпека".

Утиліта не потребує установлення. Досить запустити файл procexp.exe. Відкриється головне вікно утиліти (рис. 5.5):

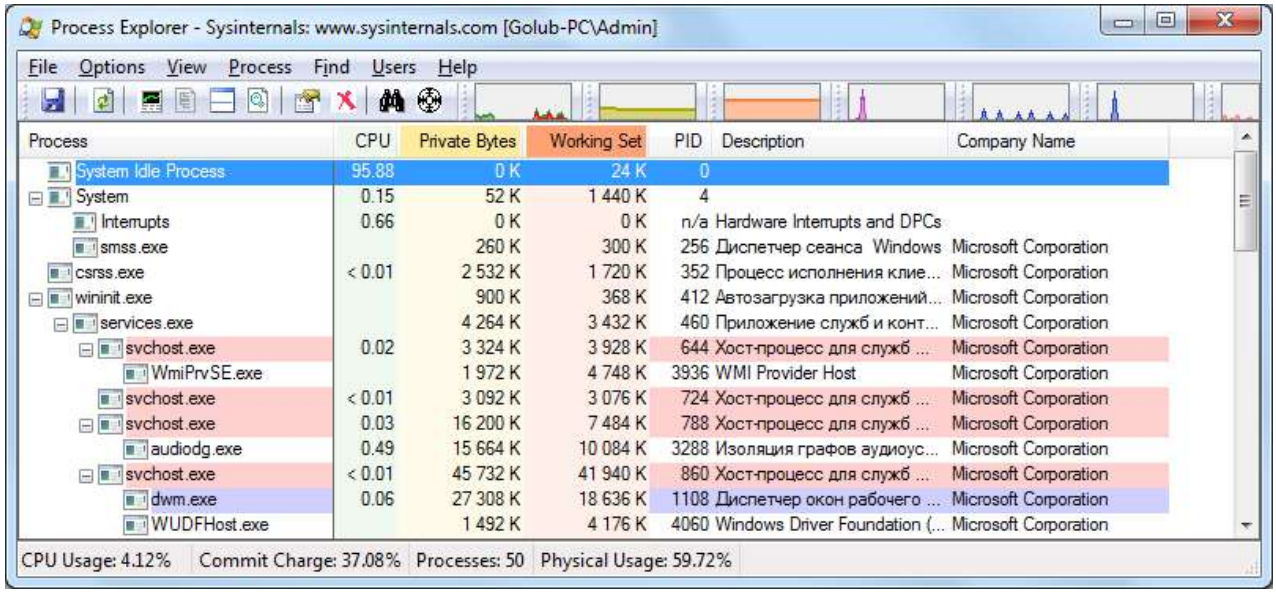


Рис. 5.5. **Головне вікно програми Process Explorer**

У верхній частині вікна в деревовидній структурі перелічено всі процеси, що працюють у системі. Окрім імені процесу, виводиться інформація про використання процесора, опис процесу, об'єм зайнятої пам'яті. Подвійне клацання по імені процесу відкриває вікно його властивостей, приклад якого показано далі (рис. 5.6).

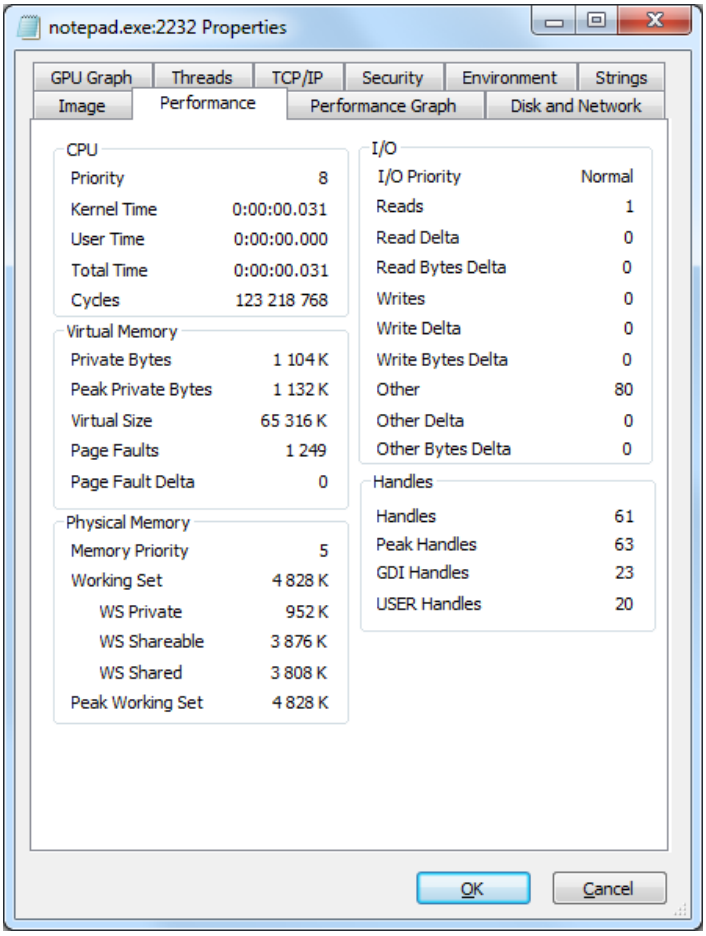


Рис. 5.6. **Вікно властивостей процесу у програмі Process Explorer**

У нижній частині головного вікна показано детальну інформацію про виділений у верхній частині процес. Це може бути список бібліотек, використовуваних процесом, або список дескрипторів (відкритих файлів, ключів реєстру тощо). Подвійне клацання мишки на ім'я бібліотек або дескриптора відкриває вікно властивостей. Для отримання опису бібліотеки за допомогою клацання правої кнопки мишки по її імені можна відкрити меню, вибір у якому пункту Google відправляє запит у пошукову систему з ім'ям вибраної бібліотеки.

Для будь-якого процесу за допомогою клацання правої кнопки мишки по імені процесу можна змінити його пріоритет, завершити його виконання або завершити виконання процесу та його дерева. Корисна особливість Process Explorer полягає в тому, що роботу будь-якого процесу можна припинити (*suspend*), а потім відновити (*resume*). Припинення роботи процесу може тимчасово звільнити зайняті ним ресурси (процесор, мережу, жорсткий диск) для використання іншими додатками. Припинивши виконання процесу можна визначити, наприклад, який процес відкриває вікна

або виявляє підвищену мережеву активність. Це буває необхідно зробити за підозри на зараження комп'ютера вірусом або для припинення роботи рекламного програмного забезпечення, яке встановлюється на комп'ютер без відома користувача.

Process Explorer надає в розпорядження користувача зручний інструмент, за допомогою якого дуже просто визначити те, яким процесом відкрито певне вікно. Для цього слід перетягнути з панелі інструментів Process Explorer кнопку в будь-яке місце вікна, що відкрилося. Після цього у верхній частині головного вікна Process Explorer буде підсвічуватися ім'я шуканого процесу.

Process Explorer може вивести приховане вікно додатка на екран. Деякі програми не відновлюють свій значок у tray, і якщо оболонка з якихось причин перезавантажується, то після цього доступ до інтерфейсу згорнутою у tray програми дістати буде неможливо. Доведеться завершувати її *Диспетчер завдань* і запускати наново із втратою всіх незбережених даних. Скориставшись пунктом контекстного меню Bring to Front, можна дістати доступ до такого вікна і зберегти дані.

Окрім іншого, за допомогою Process Explorer можна вивантажити в текстовий файл список усіх процесів з описами й об'ємом зайнятої кожним із них пам'яті, запустити будь-який додаток, вимкнути, перезавантажити або заблокувати комп'ютер, знайти використовувані бібліотеки та дескриптори, увімкнути підсвічування кольором певних процесів тощо. Усе це допомагає контролювати процеси, що працюють та отримувати вичерпну інформацію про використання системних ресурсів.

Process Monitor є вдосконаленим інструментом відстежування процесів для Windows, який у режимі реального часу відображає активність файлової системи, реєстру, а також процесів і потоків (рис. 5.7).

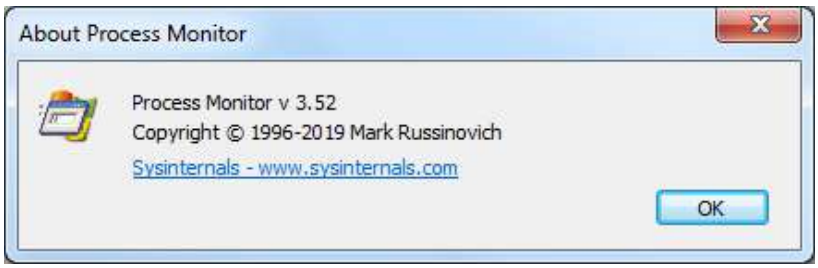


Рис. 5.7. **Діалогове вікно про програму Process Monitor**

У цій програмі поєднано такі можливості (рис. 5.8):

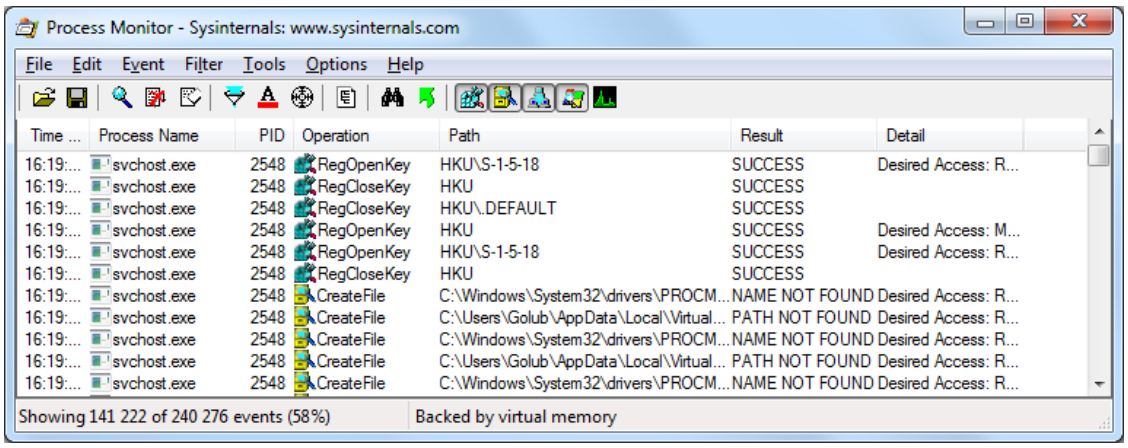


Рис. 5.8. **Головне вікно програми Process Explorer**

відстежування запуску та завершення роботи процесів і потоків, включаючи інформацію про код завершення;

відстежування завантаження образів (бібліотек DLL і драйверів пристроїв, що працюють у режимі ядра);

удосконалена архітектура запису журналів розширює можливості програми до десятків мільйонів зареєстрованих подій і гігабайтів записаних даних про події;

більше зібраних даних про параметри операцій уведення і виведення;

нешкідливі фільтри дозволяють установлювати фільтри, які не будуть призводити до втрати даних;

збирання стеків потоків для кожної операції дозволяє здебільшого визначити початкову причину виконання операції;

достовірне збирання інформації про процеси, включаючи шлях до образу процесу, командний рядок, а також ID-користувача та сесії;

колонки, що налаштовуються і переміщуються, для кожної властивості події;

фільтри можна встановити на будь-яке поле з даними, включаючи поля, які не є колонками;

дерево процесів відображає відносини між усіма процесами, перелічениими у відомостях трасування;

основний формат журналу зберігає всі дані, щоб їх можна було завантажити в іншому екземплярі програми Process Monitor;

підказування до процесів для простого перегляду інформації про образ процесу;

детальні підказування дозволяють дістати зручний доступ до форматованих даних, які не поміщаються в колонці;

пошук, що припиняється;

запис у журнал усіх операцій під час завантаження системи.

Найпростіше ознайомитися з можливостями програми Process Monitor, прочитавши файл довідки та спробувавши скористатися кожним пунктом меню й налаштування програми на робочій системі.

Щодо аналізу процесів Process Monitor дає таку унікальну можливість, як надання сумарної динамічної інформації у графічному вигляді за весь період виконання процесу. Наприклад, для процесу WinTest.exe вона буде мати такий вигляд (рис. 5.9):

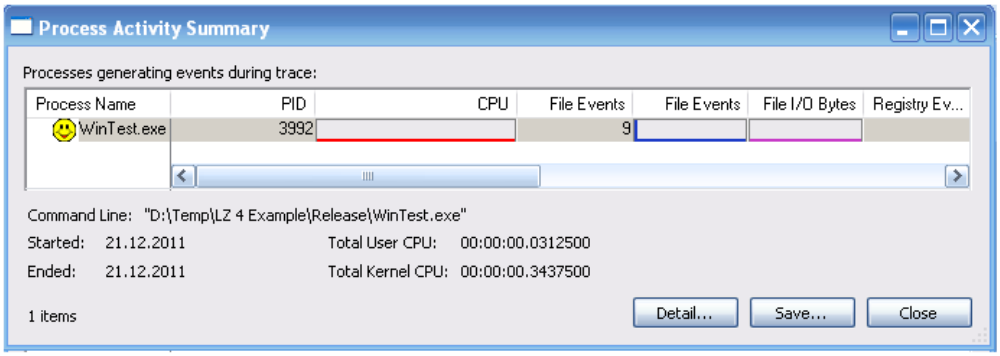


Рис. 5.9. **Аналіз властивостей процесу у Process Explorer**

Причому деталізація цих даних дозволяє оцінити ступінь впливу процесу на операційну систему (рис. 5.10):

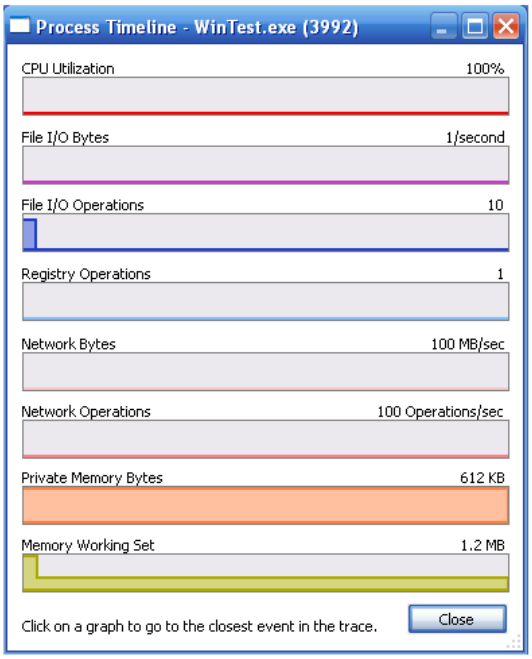


Рис. 5.10. **Детальна інформація властивостей процесу у програмі Process Explorer**

System Info for Windows (SIW) – це компактна програма, що видає докладну інформацію про апаратні засоби комп'ютера, установлене на ньому програмне забезпечення, а також мережеву інформацію. За своїми можливостями SIW аналогічна популярним програмам подібного роду, але на відміну від них має абсолютно простий інтерфейс без надмірностей, а також має безкоштовну версію для некомерційного використання (рис. 5.11).

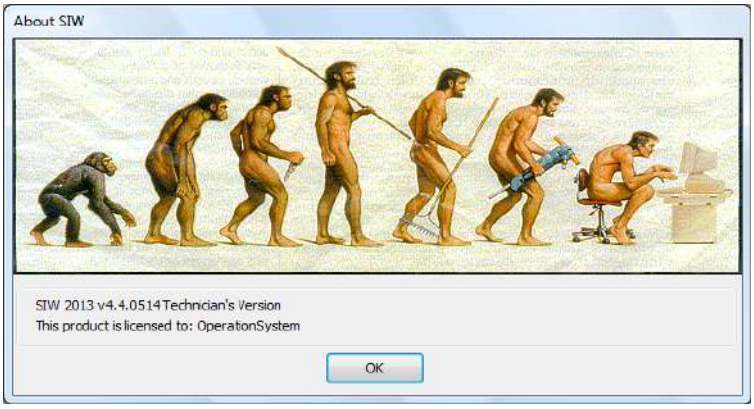


Рис. 5.11. **Діалогове вікно про програму System Info for Windows**

SIW відображає інформацію про операційну систему, оновлення, системні папки та файли, установлені програми; запущені процеси в детальному вигляді, надаючи список усіх використовуваних файлів вибраного процесу; драйверах і службах, установленому обладнанні, включаючи інформацію про материнську плату, процесор, BIOS тощо. За виданими даними можна створити файл звіту у форматах CSV, HTML, TXT або XML. Інтерфейс програми перекладено на багато мов.

Інтерфейс програми містить інтерфейс, розподілений на 3 категорії: "Обладнання", "Програми", "Мережа".

Категорія "Обладнання". SIW надає користувачам докладну інформацію про кожен елемент обладнання, зокрема процесор, PCI, мережеві адаптери, материнську плату, BIOS, пам'ять, відеопристрої, принтери, сенсори, оптичні та жорсткі диски, системні слоти, електроживлення, звукові пристрої, ресурси та багато про що інше.

Категорія "Мережа". Утиліта здатна здійснити широкомасштабний аналіз мережі та відобразити детальні відомості про неї, зокрема про відкриті порти, мережеве оточення, загальний доступ до файлів, видалених підключеннях і багато про що інше.

Категорія "Програми". Крім усього іншого, SIW робить розширений пошук із подальшим виведенням інформації на дисплей на наявність установленого програмного забезпечення та компонентів у системі, зокрема про операційну систему, установлені оновлення і програми, завантажені та DLL, системні папки та файли, драйвери, ліцензійні ключі до програм, автозавантаження, збереження паролів на web-серверах, спеціальні можливості, змінні оточення, бази даних, регіональні налаштування, типи файлів, відкриті файлі, запущені процеси, захищені файли, ActiveX і багато про що інше (рис. 5.12).

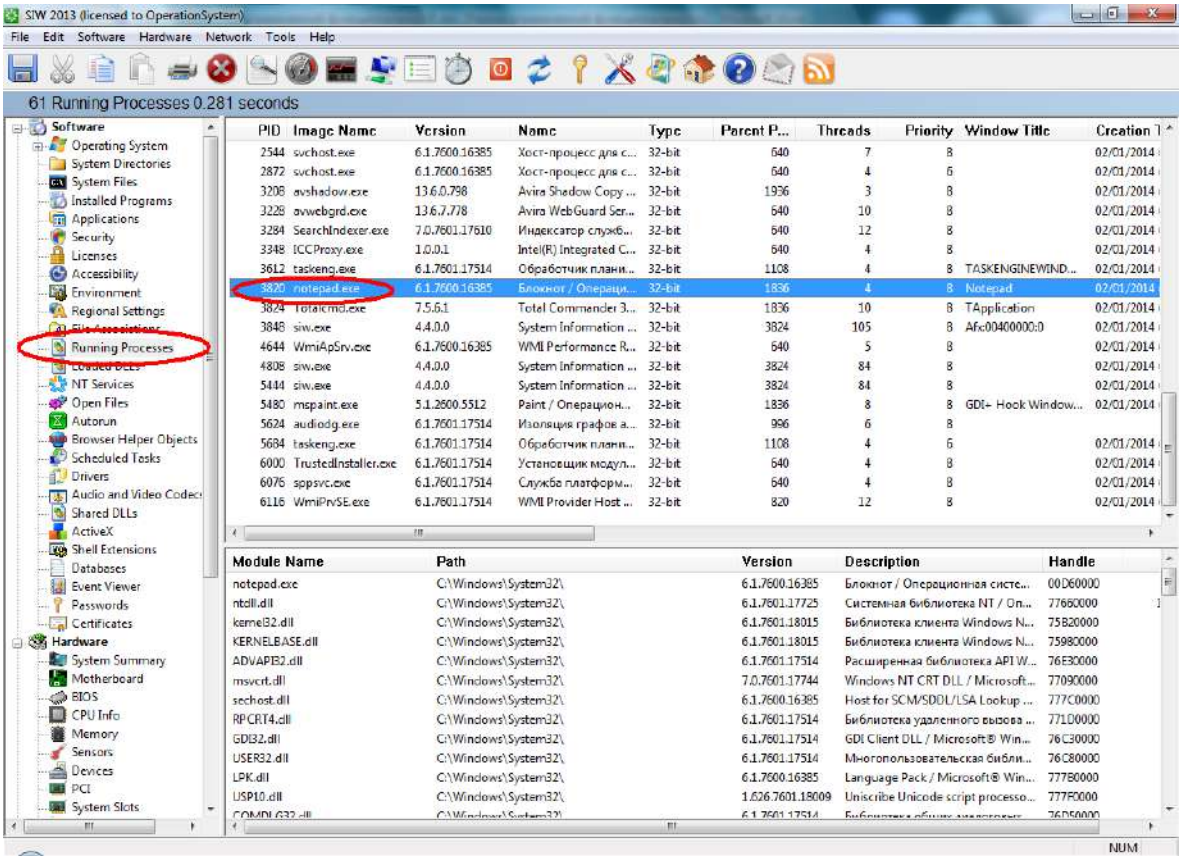


Рис. 5.12. **Головне вікно програми System Info for Windows**

TaskInfo – це програма для повного моніторингу в масштабі реального часу всіх системних процесів. Дозволяє відстежувати не тільки використання пам'яті та CPU, але й відкриті файли, використовувані бібліотеки тощо (рис. 5.13). Окрім цього, дозволяє змінювати пріоритет будь-якого процесу, а також зупиняти його або запускати. Результати моніторингу можна відстежувати візуально як у графічному режимі, так і в текстовому (рис. 5.14).

TaskInfo повністю зможе замінити зв'язку зі стандартного *Диспетчера завдань* та інструменту системної інформації.

Рис. 5.13. **Діалогове вікно про програму TaskInfo**

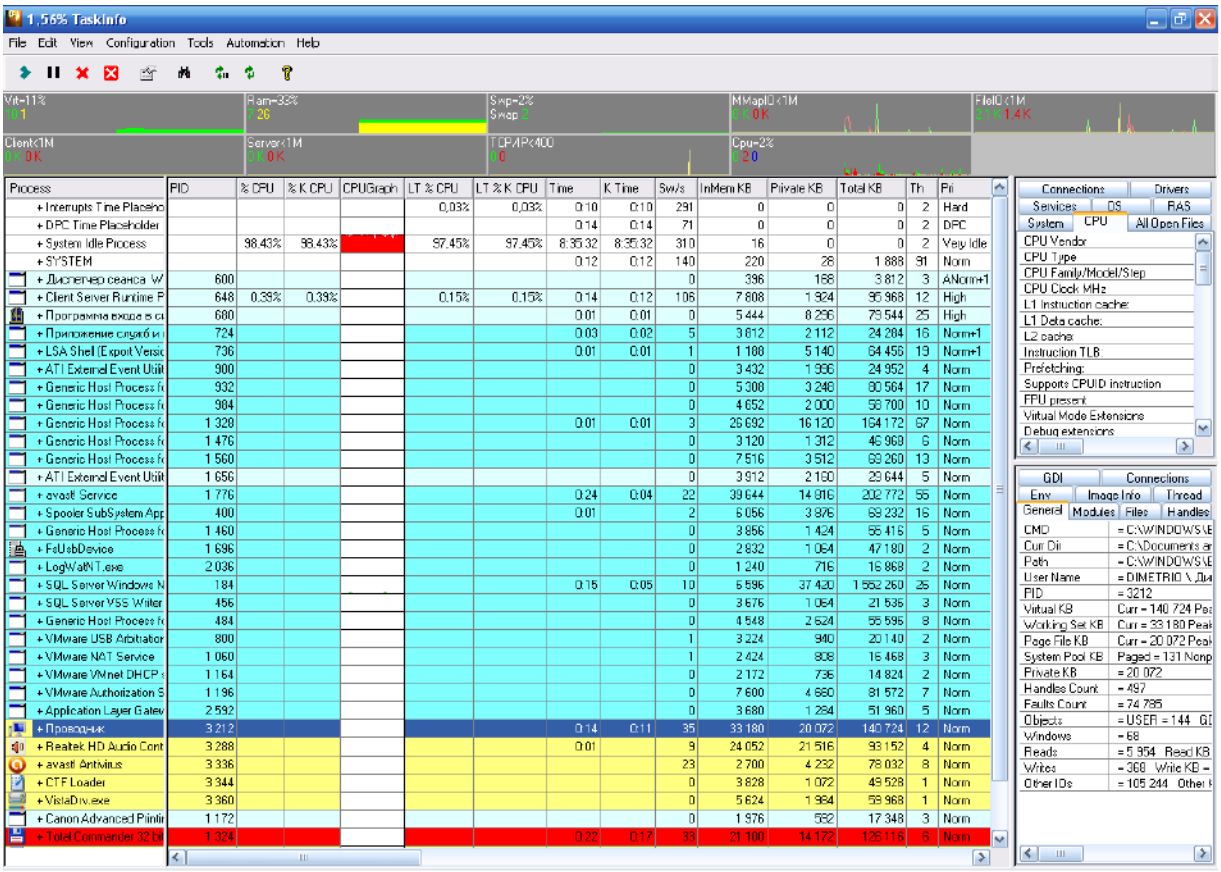


Рис. 5.14. **Головне вікно програми TaskInfo**

TaskInfo дозволяє отримати такі дані:

список усіх запущених процесів і потоків (зокрема системних потоків), із докладною інформацією про кожен процес: використання процесора і пам'яті, шлях, усі відкриті файли та модулі (DLL), параметри командного рядка, змінні оточення, відкриті з'єднання і про багато що інше;

список більшості процесів, які намагаються бути невидимим, наприклад, такі як черв'яки, key logs та інше шпигунське програмне забезпечення;

використання процесора(ів), пам'яті (фізичної, віртуальної, кеша і swap);

докладна інформація про встановлені процесори й операційну систему;

інформація про дані на локальних дисках, мережевих тощо;

усі відкриті файли, драйвери та TCP/IP, VPN-з'єднання зі всіма подробицями.

Зручна програма, яку можна використовувати як альтернативи Task Manager та інструментів System Information. TaskInfo відстежує всі запущені в системі процеси, спостерігає за завантаженістю процесора або процесорів, використанням оперативної пам'яті, відкритими файлами та шляхами до них, завантаженими бібліотеками DLL, діями, які виконують із командного рядка, мережевими з'єднаннями та багато за чим іншим.

Окрім цього, TaskInfo також відображає докладну інформацію про операційну систему, папку Windows, процесор (зокрема про підтримку MMX/SSE, тип, частоту, наявності FPU та ін.).

Додаткові можливості програми TaskInfo:

запуск або переривання виконання процесів;

зміна пріоритету процесів;

швидкий пошук інформації про процес у пошуковій системі Google;

запуск наладника процесів;

звільнення фізичної оперативної пам'яті.

Microsoft Spy++ – це службова програма на основі Win32, що надає графічне зображення системних процесів, потоків, вікон і повідомлень вікон (рис. 5.15).

Є дві версії Spy++. *Першу версію* з ім'ям Spy++ (*spyxx.exe*) призначено для відображення повідомлень, відправлених до вікна, що працює у 32-розрядному процесі.

Наприклад, Visual Studio виконується у 32-розрядному процесі. Тому Spy++ можна використовувати для відображення повідомлень, відправлених до відповідного додатка. Оскільки конфігурацію більшості версій

Visual Studio за замовчування призначено для 32-розрядного процесу, ця перша версія Spy++ є єдиною доступною в меню Інструменти в Visual Studio.

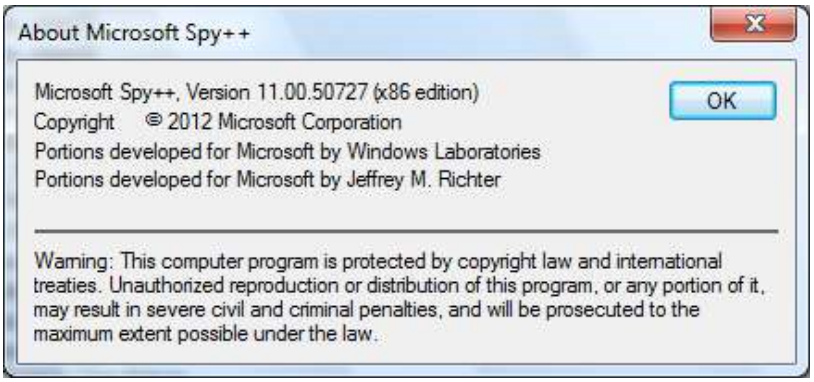


Рис. 5.15. **Діалогове вікно про програму Spy++**

Другу 64-розрядну версію з ім'ям *Spy++* (spyxx_amd64.exe) призначено для відображення повідомлень, відправлених до вікна, що працює в 64-розрядному процесі (рис. 5.16).

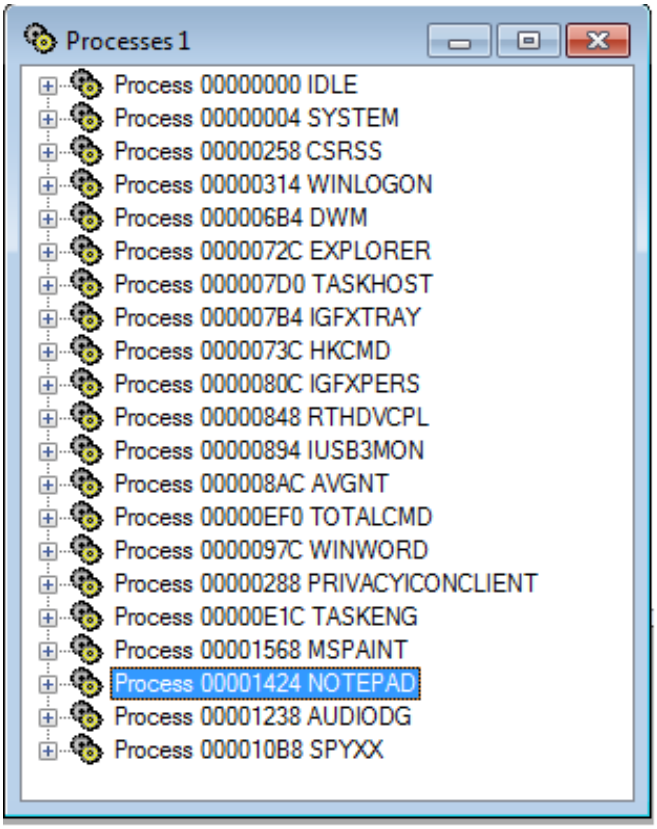


Рис. 5.16. **Вікно процесів у програмі Spy++**

Наприклад, в 64-розрядній операційній системі програма *Блокнот* виконується в 64-розрядному процесі. Звичайний шлях до Spy++ (64-розрядна версія):

..\ папка установки Visual Studio\Common7\Tools
spyxx_amd64.exe.

Будь-яку з версій Spy++ можна запустити прямо з командного рядка. Вікна зображено у вигляді дерева. Головним вікном є DeskTop, від нього йдуть відгалуження. У кожному вікні є ще вікна, і так до елементів управління.

Spy++ дозволяє виконувати такі завдання:

відображати у вигляді дерева зв'язки між системними об'єктами, включаючи процеси, потоки та вікна;

переглядати властивості вибраних вікон, потоків, процесів і повідомлень;

вибирати вікна, потоки, процеси та повідомлення безпосередньо в зображенні;

використовувати інструмент пошуку для вибору вікна покажчиком мишки;

налаштовувати параметри повідомлень шляхом вибору різноманітних параметрів, керівників записом повідомлень у журнал.

Загальне завдання для виконання

Виконайте, якщо необхідно, завантаження системного (призначеного для користувача) процесу, указаного в табл. 5.8 індивідуального завдання, дослідженню властивостей якого і буде присвячено лабораторну роботу.

До закінчення роботи індивідуальний процес
НЕ ВИВАНТАЖУВАТИ з пам'яті

Етап 1. Проведіть дослідження індивідуального процесу за допомогою програми *Диспетчера завдань*. Водночас установіть для нього базовий пріоритет, указаний у табл. 5.8 індивідуального завдання. Результати зафіксуйте у вигляді табл. 5.2. Виконайте підтвердження результатів відповідними скриншотами.

Таблиця 5.2

Результати дослідження процесу в *Диспетчерові завдань Windows*

Ім'я образу	PID	Базовий пріоритет	Максимальний об'єм фізичної пам'яті процесу	Об'єм віртуальної пам'яті процесу

Етап 2. Проведіть дослідження індивідуального процесу за допомогою програми Aida. Результати зафіксуйте у вигляді табл. 5.3. Виконайте підтвердження результатів відповідними скриншотами.

Таблиця 5.3

Результати дослідження процесу у програмі Aida

Ім'я процесу	Файл процесу	Зайнято пам'яті	Зайнято підкачуванням

Етап 3. Проведіть дослідження індивідуального процесу за допомогою програми Process Explorer. Результати зафіксуйте у вигляді табл. 5.4. Якщо у процесу декілька потоків, то перелічіть усі їхні TID. Виконайте підтвердження результатів відповідними скриншотами.

Таблиця 5.4

Результати дослідження процесу у програмі Process Explorer

Процеси	PID	Базовий пріоритет	Максимальний об'єм фізичної пам'яті процесу	Об'єм віртуальної пам'яті процесу	Дескрипторів		TID	Динамічний пріоритет
					системних	users		

Етап 4. Проведіть дослідження індивідуального процесу за допомогою програми System Info for Windows. Результати зафіксуйте у вигляді

табл. 5.5. Виконайте підтвердження результатів відповідними скриншотами.

Таблиця 5.5

Результати дослідження процесу у програмі System Info for Windows

PID	Ім'я образу	Версія	Ім'я	PID батька	Потоки	Пріоритет	Назва вікна	Час створення	Час запуску	Розмір	Ім'я файла і шлях

Етап 5. Побудуйте за допомогою програми Process Monitor дерево процесів у вашій операційній системі, причому особливу увагу приділіть дослідженню дерева для індивідуального процесу. Наприклад, для процесу Wintest.exe дерево буде мати такий вигляд (рис. 5.17):

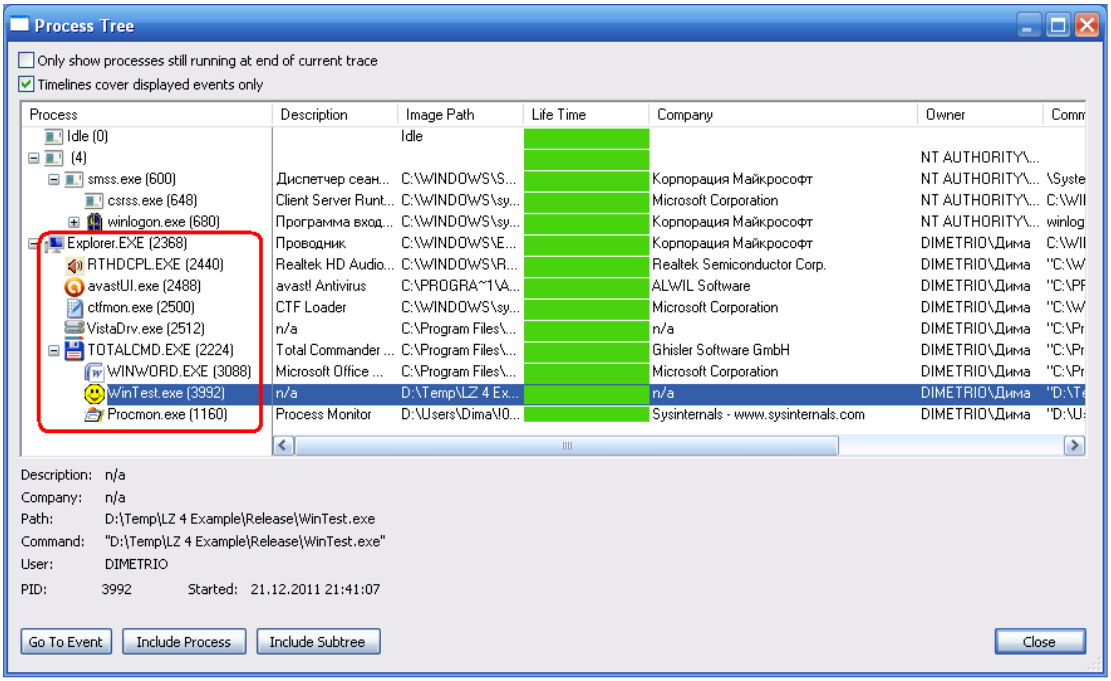


Рис. 5.17. **Дерево процесів у програмі Process Monitor**

Етап 6. Проведіть дослідження індивідуального процесу за допомогою програми Task Info. Результати зафіксуйте у вигляді табл. 5.6. Якщо у процесі декілька потоків, то перелічіть усі їхні TID. Виконайте підтвердження результатів відповідними скриншотами.

Таблиця 5.6

Результати дослідження процесу у програмі Task Info

Process	PID	Memory			TID	Priority	Handles	Windows	Version	Company	Signer
		InMem	Private	Total							

Етап 7. Проведіть дослідження індивідуального процесу за допомогою програми Microsoft Spy++. Результати зафіксуйте у вигляді табл. 5.7. Виконайте підтвердження результатів відповідними скриншотами.

Таблиця 5.7

Результати дослідження процесу у програмі Microsoft Spy++

Module Name	PID	Priority Base	Threads	Memory, KB				Page File, KB			Current Priority
				Virtual	Peak virtual	Working Set	Peak Working Set	Bytes	Peak Bytes	Page Fault	

Додаткове завдання

Створіть програмний продукт у середовищі Visual Studio, у якому слід передбачити наявність мінімум двох пунктів головного меню.

Перший пункт меню призначено для виведення статистичних даних про поточний процес (потік) у вигляді єдиного діалогового вікна таких параметрів, як:

1. Дескриптори запущеного процесу і потоку.
2. Ідентифікатори запущеного процесу і потоку.
3. Базовий і поточний пріоритет потоку.
4. Стартова адреса.
5. Стан потоку.
6. Контекст виконання потоку.

Другий пункт меню відповідає виконанню індивідуального завдання (табл. 5.8).

Таблиця 5.8

Індивідуальні завдання для виконання

Варіанти	Процеси	Базові пріоритети	Додаткові завдання
1	2	3	4
1	calc.exe	реального часу	Створіть додаток з управління рівнями пріоритетів потоків і декілька потоків із різним пріоритетом. Користувач може вибрати рівень пріоритету довільного потоку. Вибір має здійснювати в діалоговому вікні шляхом задавання певної величини слід передбачити захист від неправильно введених даних
2	clipbrd.exe	нижчий від середнього	Запуск додатка має створити шість вікон, відповідні шести потокам: одне – головне, останні – породжені, які, своєю чергою, залишаються в циклі введення аж до виходу із програми. У кожне вікно організуйте виведення унікального текстового повідомлення
3	cleanmgr.exe	вищий від середнього	Напишіть програму, яка під час натиснення мишки створює потоки: під час натиснення правої клавіші – потік, що робить виведення зростального ряду, лівою – потік зі спадним рядом. Потік вивантажується з пам'яті після закінчення рахунку. Кожному потоку поставте у відповідність своє вікно з випадковим кольором фону. Кількість потоків обмежено користувачем через контекстне меню і перебуває в діапазоні від 1 до 100
4	cliconfg.exe	середній	Створіть багатопотокову програму, яка формує потоки трьох типів. Будь-який із потоків породжено відповідним пунктом меню і захоплює, відповідно, 1, 2, 3 одиниць умовного ресурсу (максимальні числа ресурсів за замовчуванням 10 і може змінюватися користувачем у вікні діалогу, який викликають через меню). Тут під ресурсом розуміють значення глобальної змінної. Кількість, вид потоків, а також їхній стан виводиться на екран. Якщо кількість ресурсів не дозволяє працювати потоку, він перебуває у стані очікування. Видалення потоків здійснюють через меню в порядку запуску (першим видалиться потік, запущений першим)

Продовження табл. 5.8

1	2	3	4
5	magnify.exe	нижчий від середнього	Напишіть програму, що породжує 4 потоки, кожному з яких виділено четверту частину вікна додатка. Перший потік виводить у свою область зведення час створення потоку. Другий потік у випадкові інтервали часу знищує перший або третій потік, а через 0,001 – 1 с відновлює видалений потік. Третій потік заповнює свою зону вікна іконкою виконуваного модуля процесу. Четвертий потік фіксує у трьох зміни та виводить їх у своїй зоні вікна кількість запусків кожного з попередніх трьох потоків
6	mobsync.exe	реального часу	Передбачити введення користувачем як параметра роботи додатка довільного великого значення змінної. Другий пункт меню повинен мати такі підпункти: "Збільшити", "Зменшити", "Обмін". Після чого запустити декілька екземплярів їхнього додатка. Довільний вибір операцій збільшення (зменшення) приводить до зміни вибраної змінної та відображення її значення в робочій області кожного з екземплярів додатка. Крім того, там же відображено значення дескрипторів створених вікон запущених екземплярів додатків
7	mmc.exe	низький	Напишіть програму, яка під час натиснення комбінації клавіш "CTRL + F2" створює потік, що робить виведення зростального ряду в позицію курсору вікна, комбінації клавіш "CTRL + F3" – потік зі спадним рядом. Потік вивантажується з пам'яті після закінчення рахунку. Кількість потоків обмежується користувачем через контекстне меню. Кожному потоку поставте у відповідність своє вікно
8	dxdiag.exe	високий	Напишіть програму, яка спочатку перелічує всі процеси, що виконують у системі, у вигляді списку з іменами й ідентифікаторами кожного процесу. Для поточного процесу повідомляють його ідентифікатор (разом з ідентифікатором батьківського процесу), клас пріоритету і кількість потоків, що виконують зараз у контексті процесу

Продовження табл. 5.8

1	2	3	4
9	eventvwr.exe	середній	Напишіть програму мініатюрної електронної таблиці, що складається із двох комірок: WRITER і ANSWER. У комірку WRITER можна записувати довільні числові значення (зокрема і дроби), а другу – ANSWER – має бути реалізовано як статичний елемент управління, доступний тільки для читання. Поміщаючи число в поле WRITER, користувач примушує програму перелічувати значення в комірку ANSWER. Перелік полягає в тому, що лічильник, початкове значення якого дорівнює 0, поступово збільшується до максимуму, заданого в комірці WRITER. Для наочності виведіть режим, у якому перебуває програма: рахунок або очікування введення. Кожному режиму поставте у відповідність свій потік. Зміна числа в комірці WRITER у процесі рахунку має привести на початок відліку від нуля до введеного значення. Установіть крок, що дорівнює 0,1
10	netsetup.exe	низький	Запуск додатка має створити два вікна, відповідні двом потокам: головному і породженому, який, своєю чергою, залишається в циклі введення аж до виходу із програми. Коли користувач вибирає другий пункт головного меню, програма має перемикатися на приєднаний стан уведення породженого потоку, а також установити активне вікно як вікно другого потоку. Ця операція має бути успішною, якщо механізми введення обох потоків поєднано, і невдалою, якщо їх роз'єднано. Результат супроводжуйте відповідними повідомленнями в робочих областях вікон
11	cmstp.exe	низький	У програмі створіть два потоки. Призначення одного з них – періодичне читання системного часу і заповнення глобальної структури (години, хвилини, секунди), другого – виведення цієї структури на екран. Організуйте окремий доступ потоків до структури даних
12	ddeshare.exe	реального часу	Напишіть програму, яка містить два потоки. Кожному з потоків належить своє вікно і своя квітка. У меню *Налаштування* програма має здійснити виведення зображення рисунка квітів (гвоздики, троянди, ромашки тощо)

Продовження табл. 5.8

1	2	3	4
13	msiexec.exe	вищий від середнього	Створіть додатковий пункт меню *Налаштування,* у якому користувач має задати максимально допустимий час (у секундах) знаходження тестового потоку і максимальну кількість таких тестових потоків. Кожному потоку має відповідати своє вікно з випадковим кольором фону. Під час вибору третього пункту головного меню мають створювати нові потоки (і відповідні ним вікна), які через 5 с має бути завершено
14	mshearts.exe	високий	Створіть додаток з управління рівнями пріоритетів потоків і декілька потоків із різним пріоритетом. Користувач може вибрати рівень пріоритету довільного потоку. Вибір мають здійснювати в діалоговому вікні шляхом задавання певної величини. Передбачте захист від неправильно введених даних
15	charmap.exe	високий	Напишіть програму, яка має довільну кількість вікон, у яких відбувається автоматичне рисування різних геометричних фігур. Проте рисування можливе тільки тоді, коли завантажено додаток notepad із порожнім документом. Якщо завершити роботу додатка notepad, то рисування має припинитися. За подальшого запуску notepad рисування поновлюється
16	narrator.exe	нижчий від середнього	Напишіть програму, яка породжує потік із натиснення однієї з функціональних клавіш клавіатури. Кожен створений у такий спосіб потік малює фарбований круг у вікні додатка. Досягнувши межі вікна, круг змінює напрям свого проходження на протилежний. У разі руху декількох кругів пріоритет руху має бути в потоку, який створено раніше
17	osk.exe	вищий від середнього	Напишіть програму, що дозволяє експериментувати із класами пріоритетів процесів і досліджувати їхній вплив на загальну продуктивність системи. Під час натиснення на праву кнопку мишки створюється новий процес. У робочу зону вікна процесу виводиться PID процесу і його дескриптор. Спочатку перший процес у своє вікно виводить значення, що збільшується на одиницю. Одним із пунктів меню виконайте зміну класу пріоритету поточного процесу (див. табл. 5.1). У відповідне вікно виведіть повідомлення про новий пріоритет. Натиснення лівої клавіші на будь-якому із процесів приводить до його завершення

Продовження табл. 5.8

1	2	3	4
18	eudcedit.exe	вищий від середнього	Створіть додаток з управління рівнями пріоритетів потоків і декілька потоків із різним пріоритетом. Користувач може вибрати рівень пріоритету довільного потоку. Водночас вибір здійснюють за допомогою смуги прокручування. Після такого вибору виконайте функцію, що інтенсивно використовує центральний процесор. Відобразіть у робочій області здобуті результати спільно з дескриптором потоку, рівнем пріоритету
19	notepad.exe	реального часу	Напишіть програму, що дозволяє експериментувати із класами пріоритетів процесів і досліджувати їхній вплив на загальну продуктивність системи. Первинний процес постійно збільшує початкове значення довільної змінної на 1 і виводить поточне значення у вікно. Користувач, використовуючи деяке поле припинення процесу, може припинити первинний потік
20	perfmon.exe	нижчий від середнього	Напишіть програму, яка запускає новий потік від час натиснення лівої клавіші мишки. Потік починає виводити зростальну числову послідовність у робочу зону вікна. Під час натиснення лівої клавіші мишки програма видаляє потік
21	packager.exe	середній	Напишіть програму, що дозволяє експериментувати з відносними пріоритетами потоків і досліджувати їхній вплив на загальну продуктивність системи. Спочатку первинний потік працює дуже активно, і ступінь використання процесора зростає до 100 %. Усе, чим він займається, – це постійно збільшує початкове значення довільної змінної на 1 і виводить поточне значення у вікно. Передбачте зміну рівня пріоритету потоку
22	nslookup.exe	високий	Напишіть програму, яка породжує потік та натиснення однієї з функціональних клавіш клавіатури. За кожною з функціональних клавіш закріплено свій графічний об'єкт (круг, еліпс, прямокутник тощо). Кожен створений у такий спосіб потік рисує зафарбований графічний об'єкт у вікні додатка. Пріоритет під час рисування буде в того потоку, який раніше було задіяно. Таким чином, нові геометричні фігури будуть з'являтися попереду вже нарисованих

Продовження табл. 5.8

1	2	3	4
23	mstsc.exe	середній	Під час вибору другого пункту меню запускається другий процес, причому обмежено можливість подальшого запуску процесів із цього пункту. Завдання другого процесу стежити за натисненнями функціональних клавіш. У робочій області відображається назва натиснутої клавіші, а в другому вікні – факт натиснення у вигляді символу " * ". Таким чином, один процес стежить за введенням даних у другому процесі. Передбачте можливість закриття одного процесу з меню другого процесу
24	sysedit.exe	високий	Напишіть програму, яка запускає новий потік під час натиснення правої клавіші мишки. Потік починає виводити довільні числа із заданого користувачем діапазону в робочу зону вікна. Під час натиснення лівої клавіші мишки програма видаляє створений потік. Передбачте обмеження повторного натиснення правої кнопки мишки
25	taskmgr.exe	вищий від середнього	Напишіть додаток, у якому один потік буде створювати другий потік. Цей потік формує відомості про: тип процесу (64- або 32-бітовий); базовий пріоритет виконання процесу; загальну кількість потоків (threads) для кожного процесу; поточне значення номера потоку процесу. Перший потік має перебувати у стані очікування на період, що задається користувачем (від 0,001 до 50 с), після чого закриває другий потік
26	shrpubw.exe	низький	Напишіть додаток, у якому створюється мінімум чотири потоки: два – основні, два – додаткові. Основні потоки виконують у своєму вікні виведення випадкових букв, додаткові потоки – геометричні фігури (мінімум 6) випадкового кольору і розташування
27	sigverif.exe	реального часу	Коли користувач вибирає в додатку другий пункт меню, створюється потік, що виводить довільне текстове повідомлення в робочу зону вікна, яке поступово заповнює її до тих пір, поки не буде натиснуто комбінацію клавіш "CTRL + F9". Натиснення цієї комбінації має припинити виведення. Повторне натиснення має відновлювати виведення

Закінчення табл. 5.8

1	2	3	4
28	verifier.exe	середній	Створіть додатковий пункт меню *Налаштування,* у якому користувач має задати максимально допустимий час (у секундах) знаходження тестового потоку і максимальну кількість таких тестових потоків. Кожному потоку має відповідати своє вікно з випадковим кольором фону. Під час вибору третього пункту головного меню мають створюватися нові потоки (і відповідні ним вікна), які через випадковий час, але не більше від максимально допустимого, мають бути завершені
29	write.exe	нижчий від середнього	Напишіть програму, що дозволяє експериментувати із класами пріоритетів процесів і досліджувати їхній вплив на загальну продуктивність системи. Первинний процес має активно використовувати процесор, завантаживши його виконанням безлічі обчислень. Передбачте зміну класу пріоритету процесу (див. табл. 5.1). Користувач, використовуючи деяке поле *Припинення процесу,* може припинити первинний потік на задане число мілісекунд у діапазоні від 0 до 9 999
30	wscript.exe	низький	Напишіть додаток, який реалізує мультизадачний додаток MultiMDI, сенс якого полягає в тому, що в головному вікні додатка створено дочірні вікна, у яких виконують циклічне відображення еліпсів випадкової форми та кольору і реалізовано можливість установлювати відносний пріоритет окремих завдань, припиняти їх, відновлювати виконання припинених завдань, видаляти завдання

Контрольні запитання

1. Поясніть фізичний сенс поняття "базовий пріоритет процесу".

2. У чому полягають принципові відмінності програм моніторингу процесів і потоків?

3. Назвіть виробників використаних програмних продуктів.

4. Які відмітні характеристики процесів дозволяють відображати програми моніторингу?

5. Наведіть приклади використання програм у боротьбі з вірусами.

6. Обґрунтуйте переваги й недоліки програм діагностики процесів.

7. Назвіть основні характеристики процесів і потоків в Windows.

8. Хто в операційній системі задає пріоритет процесу?

9. Що таке "пріоритети реального часу"?

10. Який максимальний пріоритет має бути надано процесу ядра?

11. Назвіть тести, які використовують у програмі Aida32.

12. Поясніть фізичний уміст файла підкачування.

13. Яким чином задають ідентифікатор процесу або потоку?

14. Скільки мінімально ідентифікаторів потоків містить контейнер процесу в операційній системі Windows?

15. Що таке "дескриптор процесу"?

16. Назвіть батьківські процеси, які породжують режиму процеси користувача.

17. Поясніть призначення процесу explorer.exe в операційній системі під час побудови дерева процесів.

18. Що таке "параметр процесу Singer"? Хто його встановлює?

19. У чому полягає процедура дослідження процесів у програмі Microsoft Spy++?

20. Яким чином дескриптори вікна програми зберігаються у структурі процесу?

21. Назвіть діапазон можливих значень для ідентифікатора процесу та ідентифікатора потоку.

22. Поясніть фізичний уміст параметра Page Fault для файла підкачування.

23. У якій пам'яті зберігаються процеси та потоки?

24. Чим відрізняється фізична пам'ять процесу від його віртуальної пам'яті?

25. У якому діапазоні значень може встановлюватися динамічний пріоритет потоків, які входять до складу процесу?

26. Як називають у віртуальному адресному просторі область пам'яті зі звичайними даними, індивідуальними для будь-яких програми користувача?

27. Назвіть усі об'єкти, за яким здатна наглядати програма Spy++.

Лабораторна робота 6
Дослідження властивостей віртуальної пам'яті

Мета роботи: ознайомлення з побудовою й організацією механізму віртуальної пам'яті, дослідження станів сторінок віртуальної пам'яті, типів доступу до сторінок віртуальної пам'яті, крім того, вивчення способів отримання з купи пам'яті блоків невеликих розмірів для використання додатком.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно вивчити основні елементи організації пам'яті. Система управління пам'яттю є найскладнішою та найдосконалішою. Від того наскільки добре буде відпрацьовано програмні інтерфейси цієї системи, багато в чому залежить ефективність і працездатність створюваних додатків.

Додаткову інформацію під час підготовки до роботи можна отримати в [2; 15; 17].

Теоретичні відомості

У середовищі Windows кожен процес має власний 32-розрядний віртуальний адресний простір об'ємом до 4 ГБ. Користувачеві доступні лише 2 ГБ в області нижньої пам'яті (від 0x00000000 до 0x7FFFFFFF), а 2 ГБ в області верхньої пам'яті (від 0x80000000 до 0xFFFFFFFF) зарезервовано для ядра операційної системи.

Адреси, використовувані процесом, не відповідають фізичному розташуванню в пам'яті. Для кожного процесу ядром здійснюється перетворення віртуальних адрес у відповідні фізичні адреси. Це дозволяє *Диспетчерові пам'яті* переміщати пам'ять та управляти нею з максимально можливою ефективністю з урахуванням наявних потреб.

Адресація пам'яті здійснюється всередині адресного простору процесу. Процес не може виконувати читання/запис за межами свого адресного простору (це дозволяє захистити процеси один від одного).

Розрізняють глобальну, віртуальну пам'ять і купу.

Глобальна пам'ять розподіляється із глобальної купи процесу. Здебільшого цим дескриптором є покажчик на розподілену пам'ять. Пам'ять розподілено у вигляді приватних, виділених сторінок із доступом

для читання/запису. До приватної пам'яті не можуть дістати доступ інші процеси. Розподілені об'єкти пам'яті можуть бути нерухомими або переміщуваними. Переміщувані об'єкти можуть бути також позначені як відкидані. За допомогою віртуальної пам'яті у Win32 система може управляти пам'яттю, не впливаючи на віртуальні адреси в ній. Коли система переміщає сторінку пам'яті, віртуальна сторінка процесу просто відображається в іншому місці. Переміщувана пам'ять усе ще застосовується для розподілу відкиданої пам'яті, що використовується нечасто і лише в тих випадках, коли додаток може легко відновити вміст пам'яті.

Віртуальна пам'ять. Віртуальний адресний простір для кожного процесу набагато перевищує сумарну фізичну пам'ять, доступну для всіх процесів. Із метою збільшення розмірів пам'яті в системі для розподілу додаткової пам'яті застосовується жорсткий диск. Сумарний об'єм пам'яті, доступної для всіх процесів, є сумою об'єму фізичної пам'яті та величини вільного простору на диску, доступного для файла підкачування (файла на диску, уживаного для збільшення об'єму пам'яті). Віртуальний адресний простір організований у вигляді сторінок або одиниць об'єму пам'яті. Розмір сторінки залежить від хост-комп'ютера. На комп'ютерах x86 розмір сторінки – 4 КБ.

Оскільки віртуальна пам'ять розподіляється посторінково, її невигідно використовувати для зберігання невеликих об'єктів пам'яті.

Купа. Функції купи дозволяють процесу створювати приватну купу – блок з однієї або декількох сторінок в адресному просторі процесу. Пам'ять у приватній купі доступна тільки для процесу, що створив купу.

Для дослідження глобальної, віртуальної пам'яті та купи є дві основні групи програмних засобів: убудовані в операційну систему (ОС) і сторонніх виробників.

До *вбудованих* в операційну систему програм діагностики як фізичної, так і віртуальної пам'яті належать:

1. *Диспетчер пристроїв Windows* (Device Manager Windows).
2. *Монітор ресурсів* (Resource monitor) – для Windows 7.

До програм *сторонніх виробників*, розгляду яких присвячено лабораторну роботу, належать:

1. FreeRAM (by BySoft).
2. Mz RAM Booster v.4.1 (by Michael Zacharias).
3. FreeRAM XP Pro 1.52 (by M.Xiang).
4. CleanMem v.2.4.3 (by PcWinTech.com).

5. RAM Saver Professional v.11.11.

6. VMMap v.3.1 (by Mark Russinovich and Bryce).

7. RAMMap v.1.32c (by Mark Russinovich and Bryce).

8. Wintest.

Саме дослідженню за допомогою вказаних програм присвячено цю лабораторну роботу.

Диспетчер пристроїв використовують для оновлення драйверів обладнання, зміни налаштувань обладнання, а також для усунення неполадок (рис. 6.1).

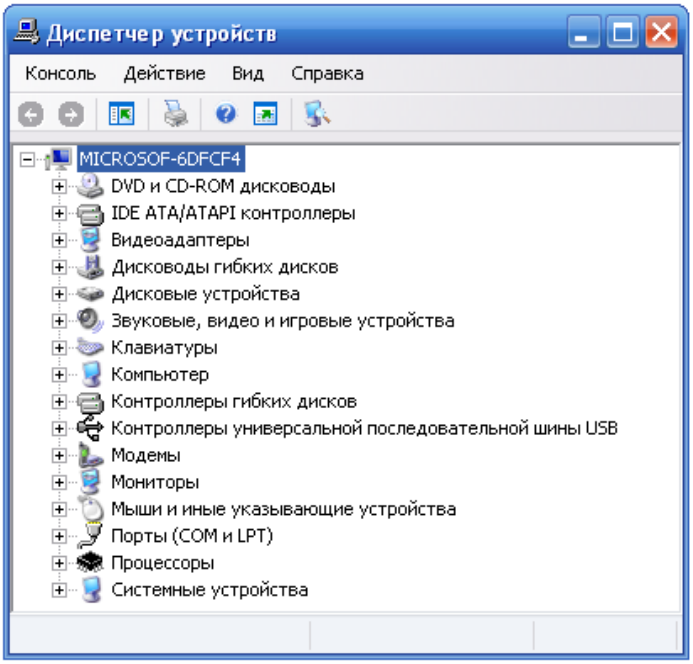


Рис. 6.1. **Вікно *Диспетчера пристроїв***

У Windows пристрої класифікують за типами. Типи пристроїв містять: плати відеоадаптерів, клавіатури, пристрої читання компакт-дисків, порти та принтери. У вікнах *Диспетчера пристроїв* і *Майстра встановлення обладнання* відображається список типів пристроїв, установлених на цьому комп'ютері.

Типи пристроїв, своєю чергою, розподіляють на категорії, відповідно до конкретних пристроїв. Наприклад, тип "Модеми" містить сотні різних модемів, які можна встановити та використовувати з Windows.

Пристрої класифікують також за способом їхнього підключення до комп'ютера. Більшість із них постійно підключено до комп'ютера та встановлено тільки один раз. Такі пристрої доступні за кожного запуску комп'ютера,

якщо тільки їх не відключено або не видалено. Прикладами можуть бути звукові плати, плати відеоадаптерів, модеми та жорсткі диски.

Інші пристрої розраховано на підключення до комп'ютера та відключення від нього в міру потреби. Їх можуть підключати до відповідного порту або гнізда розширень, і Windows розпізнає і налаштовує їх без перезавантаження комп'ютера. Під час відключення таких пристроїв необхідно лише повідомляти Windows про їхнє витягання, видалення або відключення. Вимикати або перезапускати комп'ютер не потрібно. Пристроями цього типу є, наприклад, такі:

1. Плати PC, що підключають до переносних комп'ютерів.
2. Пристрої, що підключають до шини USB або шини IEEE 1394.
3. Стикувальні вузли, що підтримують гаряче стикування і відстикування переносних комп'ютерів.
4. Пристрої, що підключають до послідовних або паралельних портів.

Монітор ресурсів (Resource Monitor). Дозволяє спостерігати за розподілом системних ресурсів між процесами та службами. Хоча у Windows 7 теж є *Монітор ресурсів,* він відрізняється від більш сучасної версії призначеним для користувача інтерфейсом і обмежений у функціональності. У Windows XP для відстежування ресурсів використовується *Диспетчер завдань* (Task Manager) (рис. 6.2).

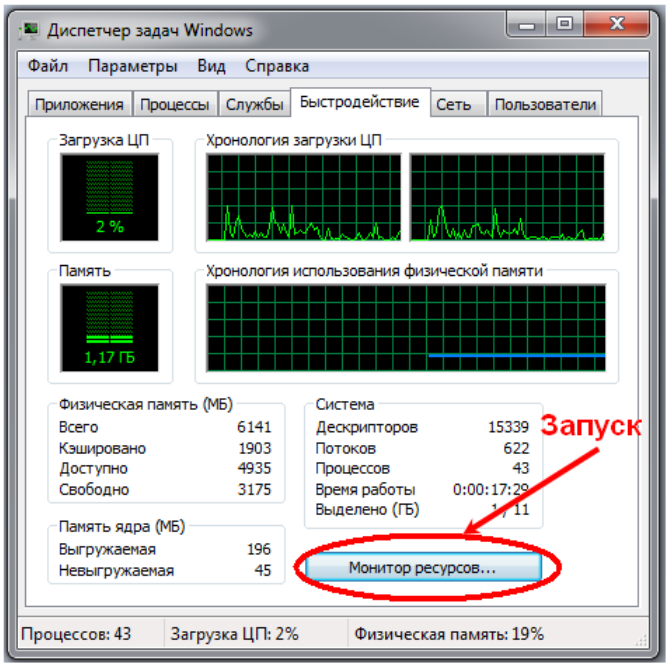


Рис. 6.2. **Вікно *Диспетчера завдань* Windows**

Запустити *Монітор ресурсів* можна декількома способами. Якщо вже відкритий *Диспетчер завдань,* перейдіть на вкладку "Швидкодія" (Performance) і натисніть кнопку "Монітор ресурсів". Його також можна викликати з меню "Пуск → Усі програми → Стандартні → Службові" або просто ввести **Resmon.exe** у рядку виконання команд і натиснути Enter.

Відкриється вікно *Монітора ресурсів* із п'ятьма вкладками. На кожній вкладці містяться численні графіки й таблиці з даними, що оновлюються в режимі реального часу (рис. 6.3).

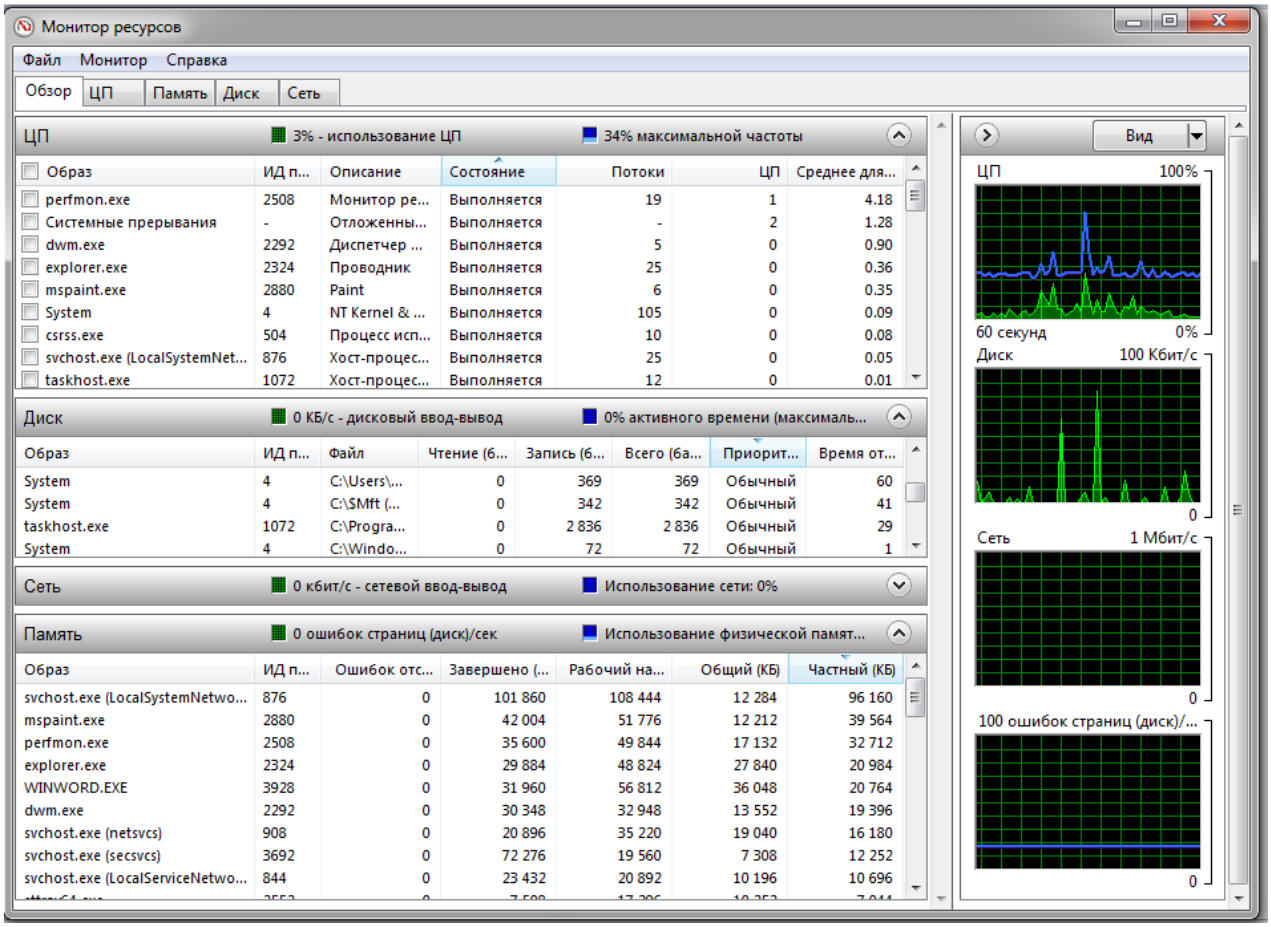


Рис. 6.3. **Вікно *Монітора ресурсів* Windows**

Із погляду лабораторної роботи цікавою буде третя вкладка *Пам'ять* (Memory). На ній наводять докладні відомості про використання пам'яті. Додатково в розділі "Фізична пам'ять" (Physical Memory) є унікальна гістограма, що відображає розподіл пам'яті (рис. 6.4).

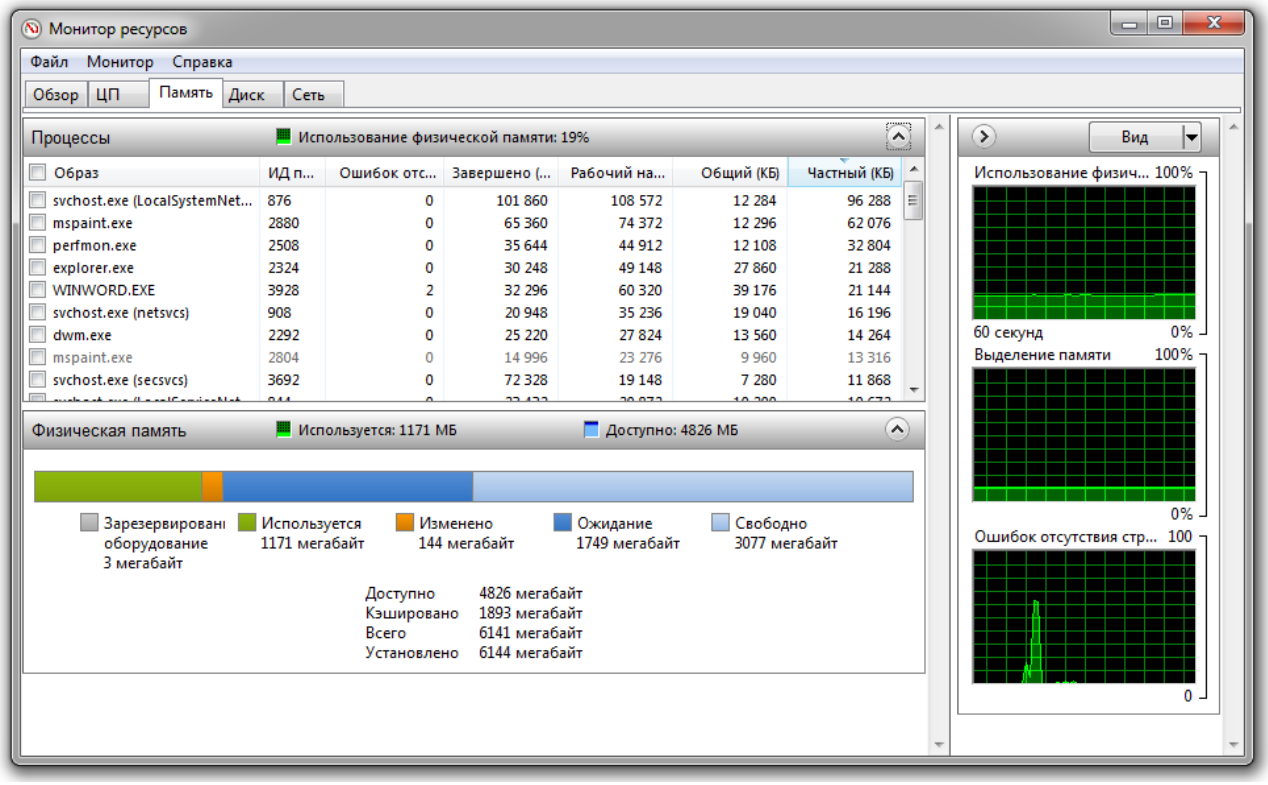


Рис. 6.4. **Розподіл пам'яті за даними *Монітора ресурсів* Windows**

Справа на кожній вкладці розташовано графіки. Вони безперервно оновлюються і відображають стан за останню хвилину. Щоб вивчити певну активність більш детально, перш ніж ця ділянка графіка зникне з очей, можна вибрати команду "Зупинити моніторинг" (Stop Monitoring) у меню "Монітор" (Monitor). Поновлюється моніторинг командою "Запустити моніторинг" (Start Monitoring).

Під час вибору певного процесу всі останні фільтруються, завдяки чому стає набагато простіше зрозуміти, як саме цей процес впливає на розподіл ресурсів.

Диспетчер пам'яті Windows створює віртуальну систему пам'яті, яка складається з доступної фізичною RAM і файла підкачування на жорсткому диску. Це дозволяє операційній системі виділяти блоки пам'яті фіксованої довжини (сторінки) із послідовними адресами у фізичній і віртуальній пам'яті.

Таблиця "Процеси". На вкладці "Пам'ять" є таблиця "Процеси" (Processes), у якій перелічено всі запущені процеси, а відомості про використовувану пам'ять розподілено на декілька категорій:

графа "Образ". У стовпці "Образ" (Image) указано ім'я виконуваного файла процесу;

графа "ІД процесу". У стовпці "ІД процесу" (PID) указано номер процесу – унікальне поєднання цифр, що дозволяє ідентифікувати запущений процес;

графа "Завершено". У стовпці "Завершено" (Commit) указано об'єм віртуальної пам'яті в кілобайтах, зарезервований системою для цього процесу. Сюди входить і використовувана фізична пам'ять, і збережені у файлі підкачування сторінки;

графа "Робочий набір". У графі "Робочий набір" (Working Set) указано об'єм фізичної пам'яті в кілобайтах, використовуваної процесом у цей момент часу. Робочий набір складається із загальної та приватної пам'яті;

графа "Загальний". У стовпці "Загальний" (Shareable) указано об'єм фізичної пам'яті в кілобайтах, яку цей процес використовує спільно з іншими. Використання одного сегмента пам'яті або сторінки підкачування для споріднених процесів дозволяє заощадити місце в пам'яті. Водночас фізично зберігається тільки одна копія сторінки, яку потім зіставляють із віртуальним адресним простором інших процесів, що до неї звертаються. Наприклад, усі процеси, ініційовані системними бібліотеками DLL (Ntdll, Kernel32, Gdi32 і User32) використовують загальну пам'ять;

графа "Приватний". У стовпці "Приватний" (Private) указано об'єм фізичної пам'яті в кілобайтах, використовуваної виключно цим процесом. Саме це значення дозволяє визначити, скільки пам'яті потрібно тому або тому додатку для роботи;

графа "Помилка відсутності сторінки в пам'яті/с". У графі "Помилка відсутності сторінки в пам'яті/с" (Hard Faults/s) указано середню за останню хвилину кількість помилок відсутності сторінки в пам'яті на секунду. Якщо процес намагається використовувати більше фізичної пам'яті, ніж доступно в цей момент, система записує частину даних із пам'яті на диск – у файл підкачування. Подальше звернення до даних, збережених на диск, і називають помилкою відсутності сторінки в пам'яті.

Таблиця "Фізична пам'ять". У таблиці "Процеси" наводять детальні відомості про розподіл пам'яті між окремими процесами, а таблиця "Фізична пам'ять" (Physical Memory) дає загальну картину використання RAM. Її ключовий компонент – унікальна гістограма.

Кожну секцію гістограми позначено власним кольором, вона показує певну групу сторінок пам'яті. У міру використання системи, Диспетчер

пам'яті у фоновому режимі переміщає дані між цими групами, підтримуючи тонкий баланс між фізичною і віртуальною пам'яттю для забезпечення ефективної роботи всіх додатків.

Секція "Зарезервоване обладнання". Зліва розташовано секцію "Зарезервоване обладнання" (Hardware Reserved), позначену сірим кольором: це пам'ять, виділена на потреби підключеного обладнання, яку воно використовує для взаємодії з операційною системою. Зарезервована для пристроїв пам'ять заблокована і недоступна Диспетчерові пам'яті. Зазвичай об'єм пам'яті, виділеного пристрою становить від 1 до 70 МБ, проте цей показник залежить від конкретної конфігурації системи та в окремих випадках може досягати декількох сотень мегабайтів. До компонентів, що впливають на об'єм зарезервованої пам'яті, належать:

BIOS;

компоненти материнської плати – наприклад, удосконалений програмований контролер переривань уведення/виведення (APIC);

звукові карти та інші пристрої, що здійснюють уведення/виведення з відображенням на пам'ять;

шина PCI Express (PCI-E);

відеокарти;

різні набори мікросхем;

флеш-накопичувачі.

Секція "Використовується". Секцію "Використовується" (In Use), позначено зеленим кольором, показує об'єм пам'яті, використовуваної системою, драйверами та запущеними процесами. Об'єм використовуваної пам'яті розраховують, як значення "Усього" (Total) за вирахуванням суми показників "Змінене" (Modified), "Очікування" (Standby) і "Вільно" (Free). Свою чергою, значення "Усього" – це показник "Установлено" (Installed RAM) за вирахуванням показника "Зарезервоване обладнання".

Секція "Змінено". Оранжевим кольором виділено секцію "Змінено" (Modified), у якій показано змінену, але незадіяну пам'ять. Фактично її не використовують, але може бути в будь-який момент задіяно, якщо знову вона знадобиться. Якщо пам'ять не використовують достатньо давно, дані переносяться у файл підкачування, а пам'ять переходить у категорію "Очікування".

Секція "Очікування". Секція "Очікування", позначена синім кольором, показує сторінки пам'яті, видалені з робочих наборів, але як і раніше

з ними пов'язані. Інакше кажучи, категорія "Очікування" – це фактично кеш. Сторінкам пам'яті в цій категорії привласнюють пріоритет від 0 до 7 (максимум). Проте всі сторінки в категорії "Очікування" доступно для запису даних від інших процесів.

Секція "Вільно". У категорії "Вільно", позначеній блакитним кольором, показано сторінки пам'яті, ще не виділені жодному процесу або ті, що звільнилися після завершення процесу. У цій секції відображено як ще не задіяну, так і вже звільнену пам'ять, але, насправді, ще не задіяна пам'ять належить до іншої категорії – "*Нульові сторінки*" (Zero Page), яку так називають тому, що ці сторінки заповнено нульовим значенням і вони готові для використання.

FreeRAM (by BySoft) – це простий Диспетчер пам'яті з маленьким розміром дистрибутиву. Інтерфейс досить простий і має декілька індикаторів (завантаження процесора, об'єм вільної пам'яті тощо), включаючи графік використання пам'яті. Диспетчер має підтримку оболонок, серед яких є оболонки невеликого розміру, це дозволяє не ховати диспетчер і відстежувати інформацію про стан пам'яті. У налаштуваннях можна встановити автоматичне звільнення пам'яті за декількома умовами. Так само можна налаштувати звільнення пам'яті гарячими клавішами і поставити звукове сповіщення після виконання звільнення пам'яті (рис. 6.5).

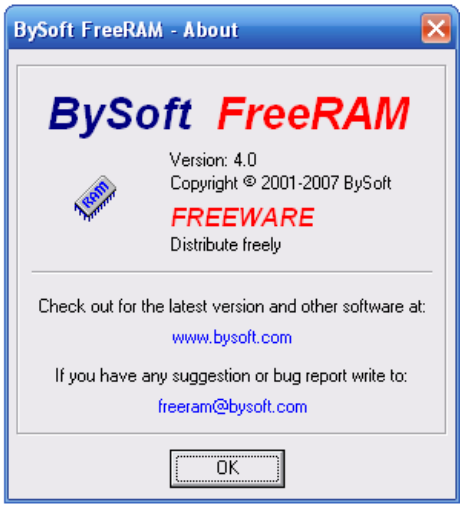


Рис. 6.5. **Діалогове вікно про програму FreeRAM**

Характеристики:
простота у використанні;
ручна й автоматична оптимізація пам'яті;

скидає на диск невикористовувані бібліотеки;

підвищує ефективність комп'ютера;

із легкістю звільняє певний обсяг пам'яті натисненням кнопки;

із легкістю звільняє заздалегідь певний обсяг пам'яті (5 %, 10 % ... 80 %);

відображення графіка вільної пам'яті та використання ЦПУ в режимі реального часу;

автоматично або вручну налаштовує системний файловий кеш;

підтримує змінні оболонки;

використовує системний tray.

Програма FreeRAM має прості налаштування (Options), згруповані у вигляді чотирьох закладок:

1. Automatic.
2. Setting.
3. Skin.
4. Cache Tuning.

Якщо опцію Automatic включено, то буде відбуватися автоматичне звільнення пам'яті, що досягає критичної позначки за значення, визначуваного параметром "Об'єм вільної пам'яті (МБ)" (Amount of memory free(MB)). Водночас кількість спроб звільнення пам'яті не буде перевищувати значення, визначуваного параметром Number of retries if desired amount is not freed (рис. 6.6).

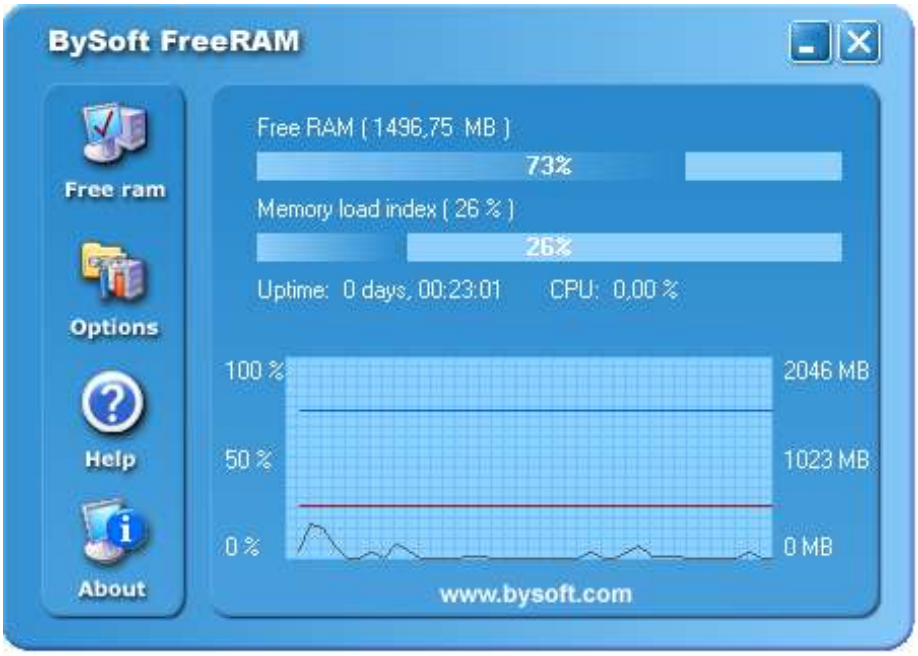


Рис. 6.6. **Головне вікно програми FreeRAM**

Друга група параметрів (Setting) визначає порядок відображення програми: зверх інших вікон (*Stay on top*), запускатися під час запуску Windows (*Start with windows*), запускатися мінімізованою у трей (*Start minimized*) тощо.

Третя група параметрів (Skin) визначає спосіб відображення самої програми за рахунок використання одного з раніше установлених скінів.

Четверта група параметрів (Cache Tuning) не працює з операційними системами Windows на базі технології NT.

Mz RAM Booster (by Michael Zacharias). Невеликий безкоштовний додаток для очищення оперативної пам'яті комп'ютера, таким чином збільшуючи швидкість роботи системи (рис. 6.7).

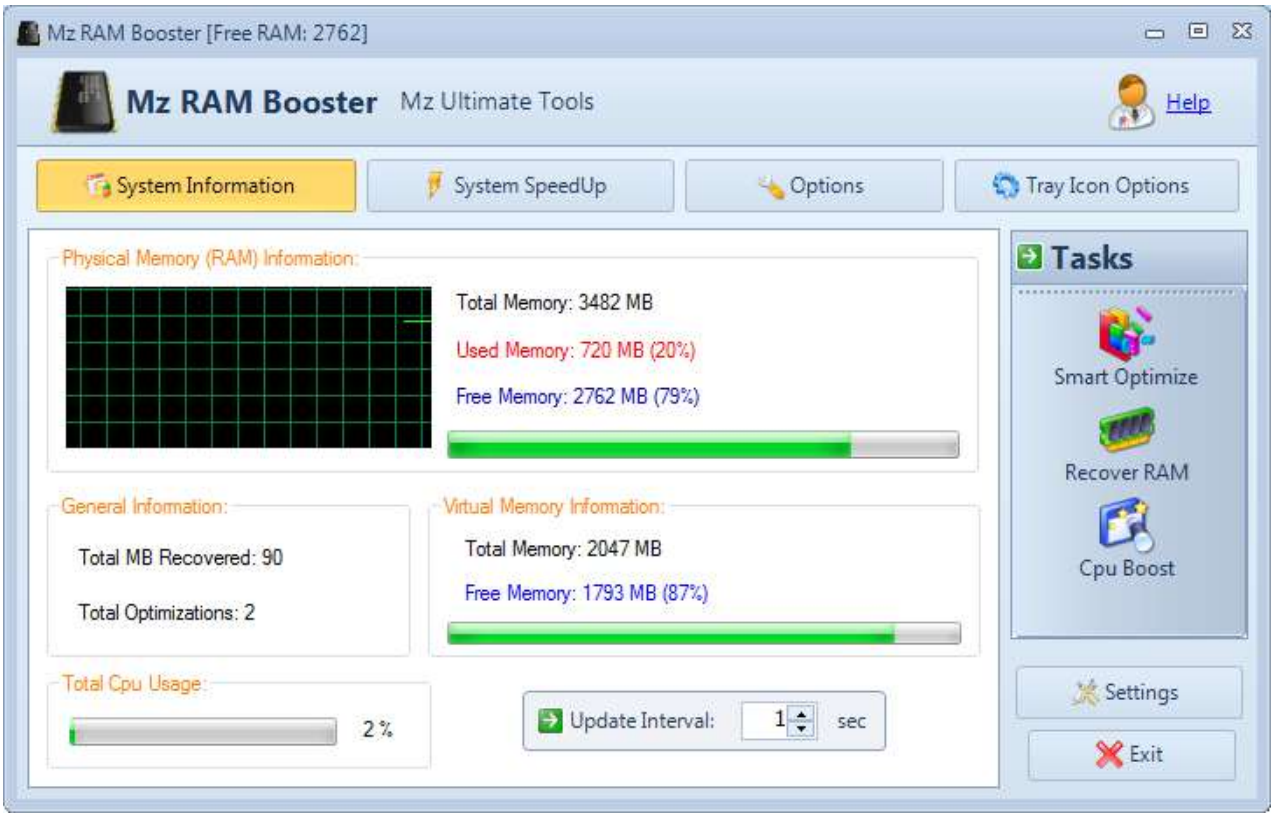


Рис. 6.7. **Головне вікно програми Mz RAM Booster**

Оперативна пам'ять комп'ютера, дійсно, важлива для швидкого виконання процесів, а звільнення пам'яті від невживаних завдань, дозволяє виграти вільний простір. Також Mz Ram Booster надає користувачеві інформацію про використання оперативної та віртуальної пам'яті. Цей засіб оптимізує роботу оперативної пам'яті за рахунок завершення процесів, що перебувають у стані простою, або зупинки фонових завдань. Користувач

зможе без ризику для стабільної роботи системи зробити роботу оптимальною, підвищивши продуктивність.

У головному вікні програми можна знайти основну інформацію про завантаженість фізичної пам'яті, а графік допоможе наочно оцінювати зміну завантаженості. Також можна побачити основні значення з використання віртуальної пам'яті, тобто розмір і зайнятість файла підкачування. Для загальної інформації автор додав у це вікно індикатор загальної завантаженості центрального процесора і можливість вплинути на інтервал оновлення даних. Для зручності у правій частині вікна реалізовано швидкі кнопки для запуску основних завдань.

Вкладку System SpeedUp призначено зробити зміни для прискорення роботи системи. Параметрів для коректування не багато, але всі вони дійсно належать до оптимізації комп'ютера, а саме можна:

заблокувати функцію Windows за створення штампа останнього часу доступу до NTFS-файлів (Disable NTFS Last Access Update Stamp);

відключити підтримку створення імен формату 8.3 (Disable 8.3 Filename Creation);

прийняти зберігання ядра системи в пам'яті (Always Keep Windows Kernel In Memory);

автоматично вивантажувати невикористовувані бібліотеки DLL (Automatic Unload Unused DLLs);

вибрати пріоритет для фонових або активних завдань (блок CPU Priority Tweaks);

визначити деякі параметри автозавершення завдань, служб і завислих додатків (блок Shutdown Tweaks).

FreeRAM XP Pro 1.52 (by M. Xiang). *Диспетчер оперативної пам'яті* комп'ютера. Ця утиліта дозволяє виконати дефрагментацію і звільнити оперативну пам'ять під час роботи комп'ютера. Тим самим збільшується швидкість і стабільність системи загалом. Слід зазначити високу швидкість роботи програми, завдяки ефективному алгоритму дефрагментації. Утиліта містить безліч налаштувань, що дозволяють налаштувати свою роботу під певні призначені для користувача вимоги. Наприклад, можна встановити параметри запуску дефрагментації, автоматичного очищення пам'яті, включити звуковий супровід подій. Тут наявний ряд інформаційних інструментів (рис. 6.8).

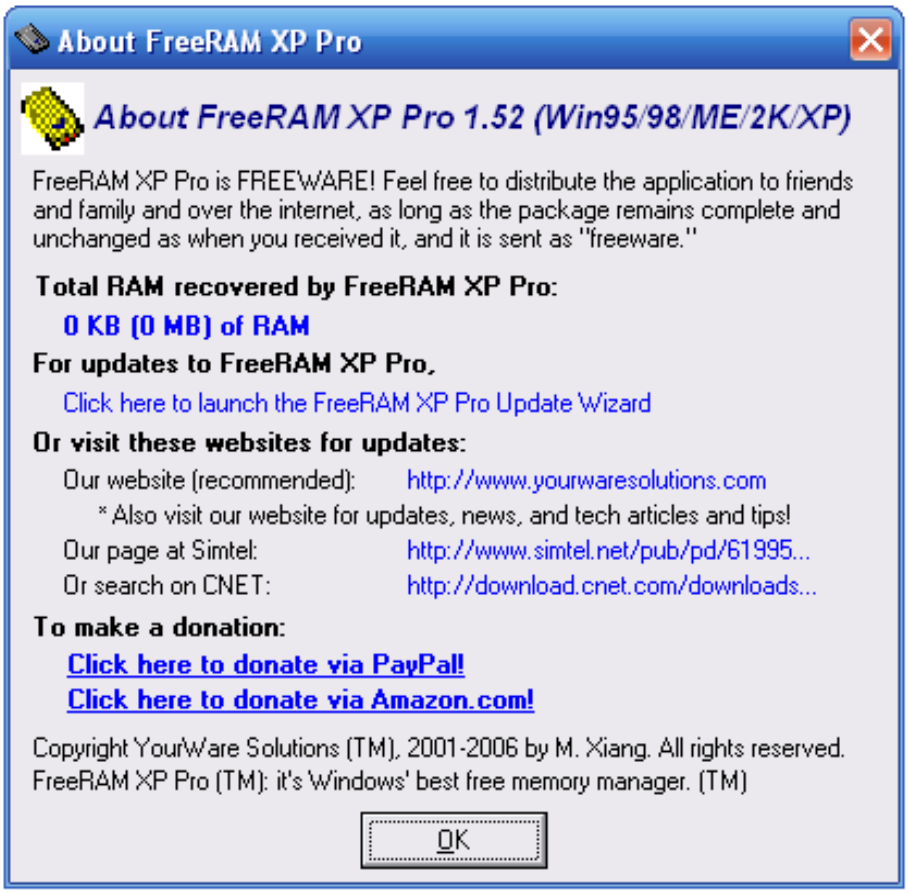


Рис. 6.8. **Діалогове вікно про програму FreeRAM**

Наприклад, інструмент Report Process Memory Usage дозволяє відобразити об'єм займаної пам'яті кожного із процесів, запущених у системі. FreeRAM XP Pro не вимогливий до програмних і апаратних ресурсів комп'ютера. Окрім двох стандартних індикаторів завантаження пам'яті, передбачено індикатори завантаження віртуальної пам'яті та процесора. Загальні відомості про основні (base) і розширені (advanced) параметри оперативної пам'яті, кеш, файла підкачування і віртуальної пам'яті містяться у звіті File → Generate Memory Report.

Працювати FreeRAM може як в автоматичному, так і ручному режимах. Відмітною особливістю програми FreeRAM XP Pro є можливість здійснити аналіз як фізичної пам'яті, так і віртуальної пам'яті процесів. Перший вид аналізу відображається у вигляді гістограми на головному вікні програми. Другий – запускається за допомогою меню Tools → Report Process Memory Usage, унаслідок чого відкривається відповідне діалогове вікно (рис. 6.9).

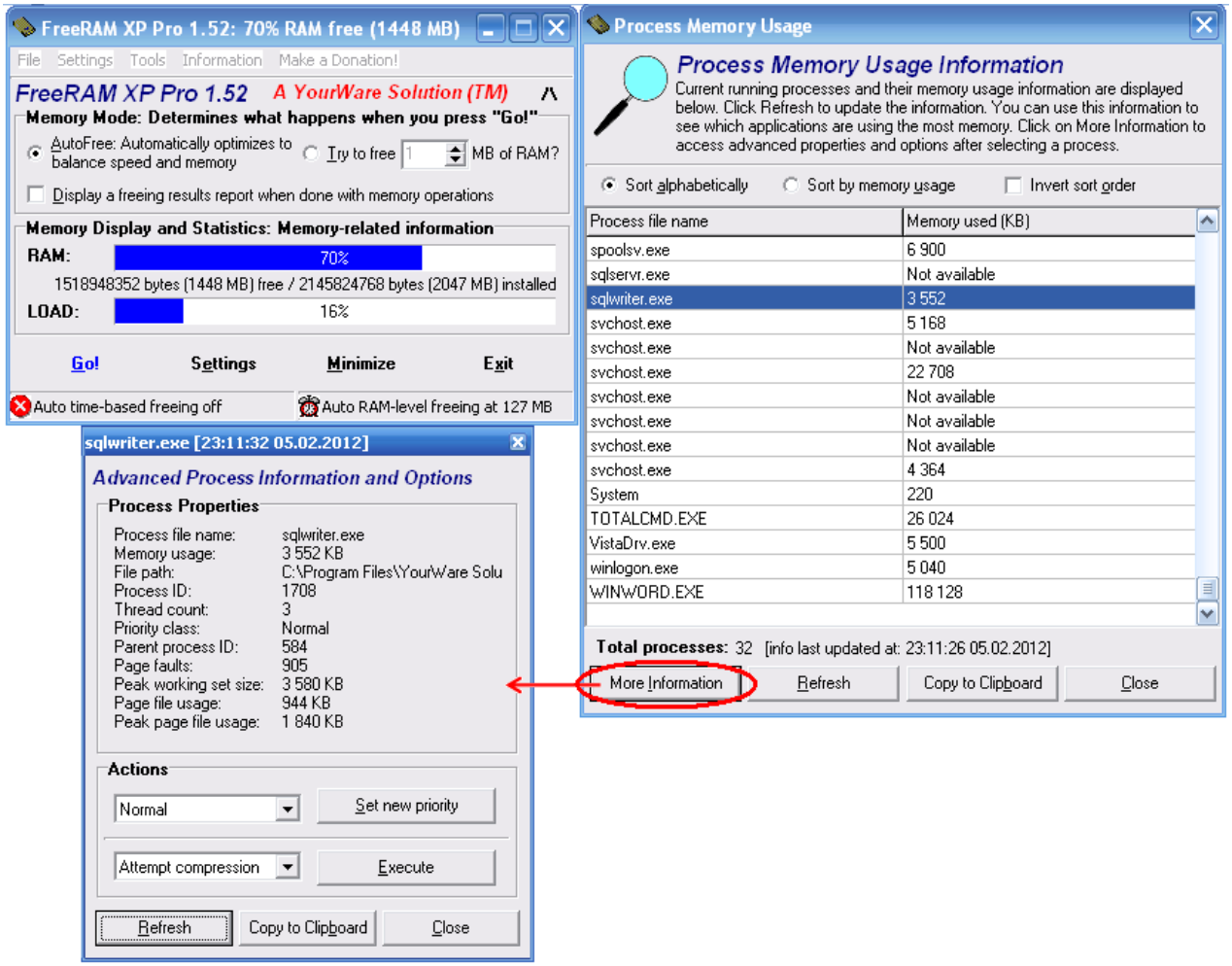


Рис. 6.9. **Головне вікно програми FreeRAM XP Pro**

Вибір процесу, що цікавить, дозволить визначити з нього розширений набір таких параметрів: ім'я файла процесу, використовуваний об'єм пам'яті, шлях до файла, ідентифікатор процесу, кількість потоків, клас пріоритету процесу, ідентифікатор батьківського процесу, кількості помилок сторінок, максимальний робочий об'єм процесу, ємність використовуваної частини процесом файла підкачування, максимальний об'єм використовуваного файла підкачування.

Утиліта підтримується всіма версіями Windows: від Windows XP до Windows 10.

CleanMem v.2.4.3 (by PcWinTech.com). Утиліта CleanMem є непоганим оптимізатором оперативної пам'яті. Актуальність використання програми полягає в необхідності у використанні на комп'ютерах користувачів із невеликим об'ємом оперативної пам'яті.

Запуск цієї утиліти може значно очистити оперативну пам'ять, тим самим приборкавши апетити досить ресурсоємних процесів, запущених

в операційній системі, буде звільнено пам'ять для інших процесів, що мають потребу (рис. 6.10).

Рис. 6.10. **Головне вікно програми CleanMem**

CleanMem достатньо проста в обігу, швидко встановлюється і вмить запускається. Управління програмою можливо як і за допомогою контекстного меню в системному Tray, так і за допомогою пункту *Menu* графічної оболонки Mini Monitor Free.

Варто зазначити, що CleanMem дуже мало важить, майже не вантажить систему, не витрачає оперативної пам'яті. Запускати програму можна на будь-якому комп'ютері, переносити із собою її можна на будь-якому накопичувачі. CleanMem запросто може виконати оптимізацію пам'яті, роблячи це за допомогою стандартного інтерфейсу Windows.

Зверніть увагу, що програма має досить гнучкі налаштування, наприклад, використовуючи спеціальні сценарії CleanMem можна легко змусити програму ігнорувати один процес і включати у процес оптимізації всі останні, тобто можна виділити ті, які чіпати не варто. Якщо буде бажання, то можна налаштувати запуск утиліти за потрібним розкладом, програма веде звіти, які можна потім переглядати.

RAM Saver Professional v.11.11. інструмент для професійного моніторингу, очищення й оптимізації оперативної пам'яті. Слугує для підвищення продуктивності операційної системи, звільняючи оперативну пам'ять від драйверів і процесів MS Windows для додатків тих, що потребують максимального завантаження процесора й оперативної пам'яті (рис. 6.11).

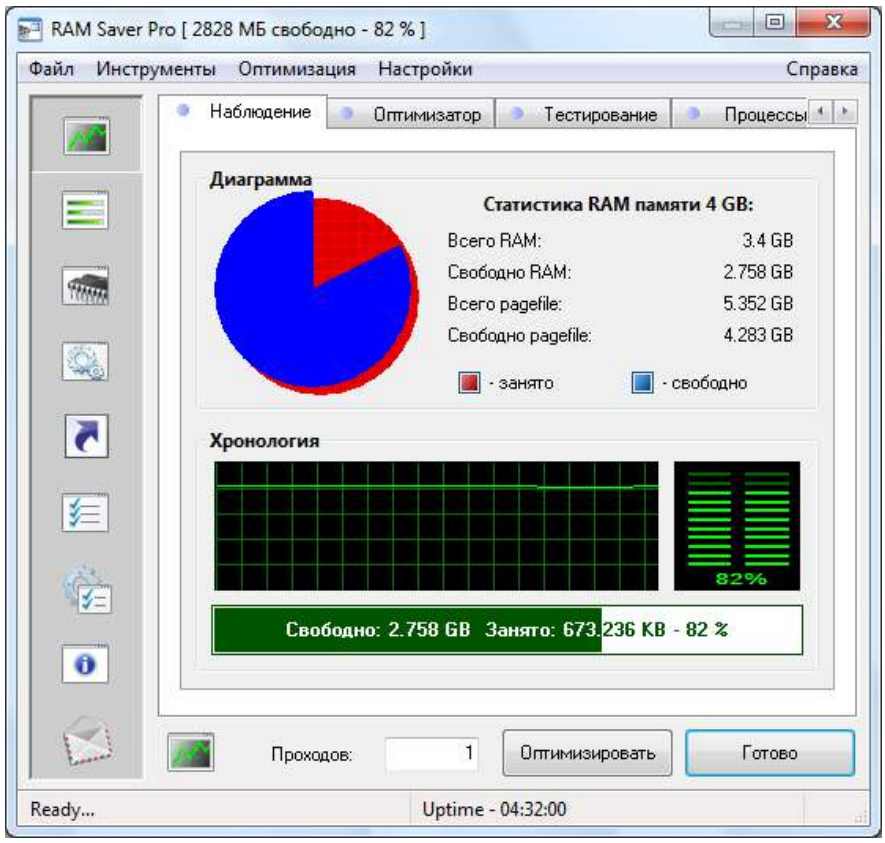


Рис. 6.11. **Головне вікно програми RAM Saver Professional**

Основні характеристики програми:
System Tray RAM монітор;
монітор робочого столу;
спеціалізована панель управління;
професійний моніторинг;
гнучка оптимізація з виведенням статистики;
RAM-тест продуктивності;
моніторинг і управління процесами що відбуваються в RAM;
можливість створення "boosted-ярликів";
основні та додаткові налаштування;
автоматична й інтелектуальна оптимізація;
швидкий виклик інструментів;
примусове очищення буфера обміну;
можливість закриття всіх додатків одним натисненням;
відображення часу з моменту увімкнення комп'ютера;
придушення та швидкий запуск зберігача екрана;
перевірка наявності компакт-диска у приводі DVD/CD;
можливість приховати всі іконки робочого столу;
примусове вимкнення і перезавантаження комп'ютера.

VMMap v.3.11 (by Mark Russinovich and Bryce). Утиліта для діагностики неполадок на основі використання системної пам'яті: дозволяє визначати візуальну карту розподілу фізичної та віртуальної пам'яті (рис. 6.12).

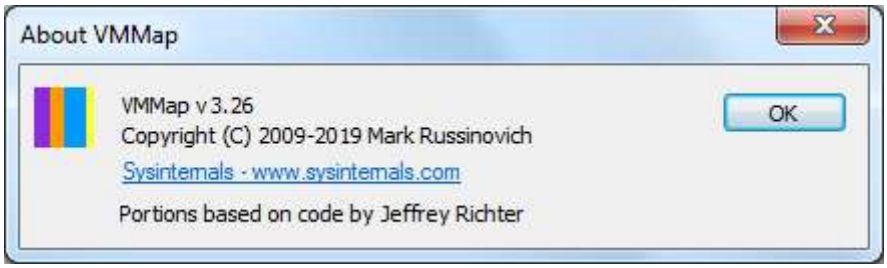


Рис. 6.12. **Діалогове вікно про програму VMMap**

Особливості програми VMMap:

1. Висока швидкість діагностики.
2. Висока надійність.
3. Програма тестує: відеопам'ять, власну пам'ять, спільно використовувану пам'ять; файл, що відображається на пам'ять; купу; керовану купу; стек і системну пам'ять.
4. Дозволяє уникнути великих поломок.
5. VMMap абсолютно безкоштовна утиліта.

Використання VMMap зовсім нескладне: після встановлення і запуску можна вибрати в меню один із запущених процесів для складання карти пам'яті. Спочатку слід дати загальну схему класифікації різних блоків в адресному просторі, щоб склалося цілісне уявлення про карту пам'яті VMMap, а потім розглянути кожен пункт окремо.

На карті буде показано різницю в об'ємі переданої пам'яті та робочого набору, а також конкретні адреси для різних категорій використовуваної пам'яті – відеопам'ять, власна пам'ять, спільно використовувана пам'ять, файл, що відображається на пам'ять, купа (heap), керована купа, стек і системна пам'ять.

Зона секції коду (Image). Її позначено у VMMap фіолетовим кольором. У ній відображається сама програма і всі використані бібліотеки. Програма з'являється в адресному просторі за допомогою механізму проектування файлів. Exe-файл проектується на адресний простір програми та стає його частиною. Будь-яке звернення до пам'яті за цими адресами приведе до фактичного читання даних із диска. Із погляду програми ця частина має вигляд звичайної пам'яті, що нічим не відрізняється

від будь-якої іншої пам'яті. Але, насправді, ця пам'ять читається з файла на диску (рис. 6.13).

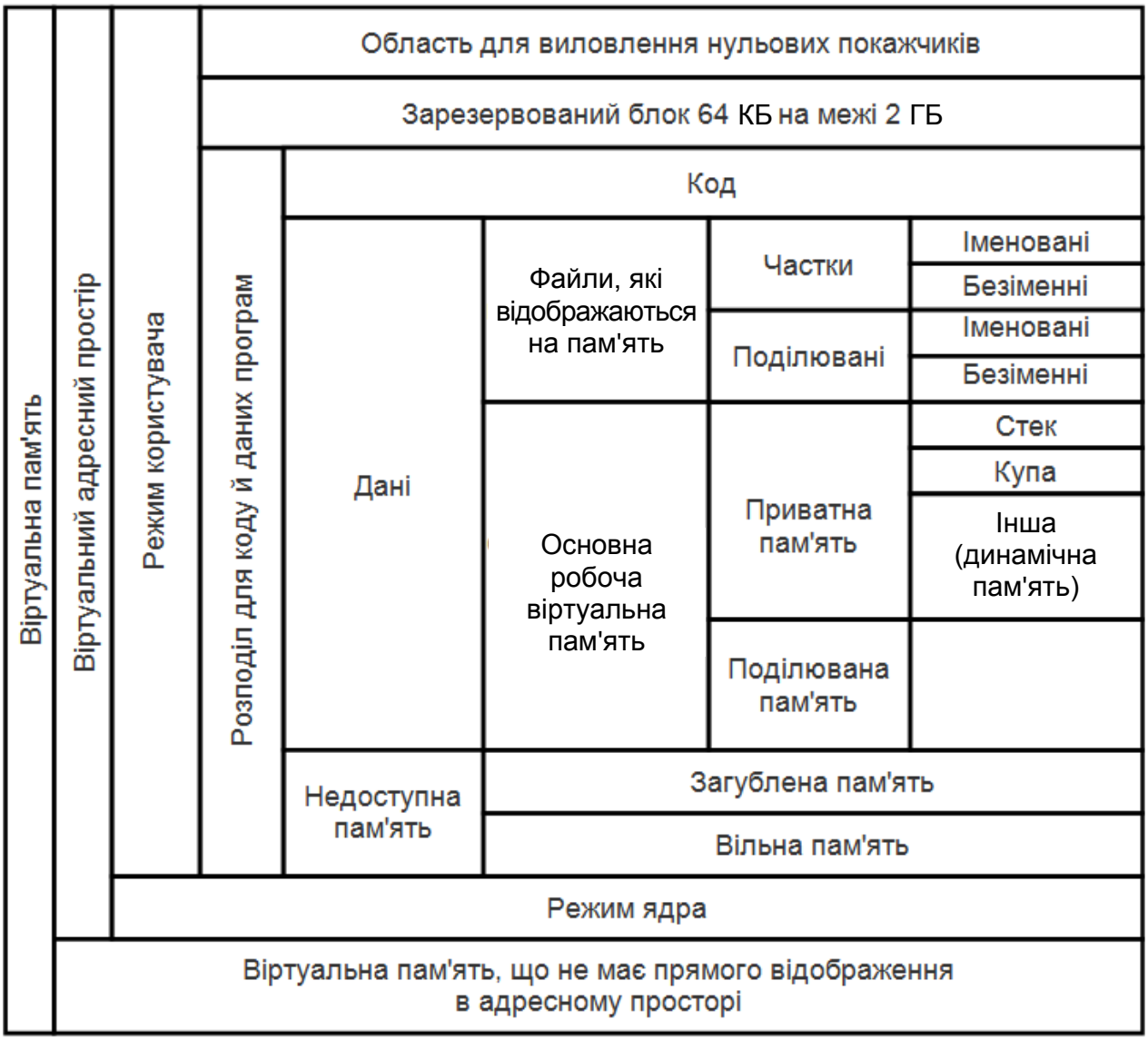


Рис. 6.13. **Структура віртуальної пам'яті, яка відображається у програмі VMMap**

Атрибути захисту (доступу) виставляють на рівні сторінок пам'яті (розмір сторінки пам'яті – 4 КБ). Дотримання режиму доступу контролюється апаратно самим процесором. Усього є три атрибути – читання, запис і виконання, які можуть комбінувати в будь-якій послідовності (відсутність атрибутів можна вважати як атрибут "No access" – PAGE_NOACCESS), а також два спеціальні модифікатори – copy-on-write і guard.

Усе, що йде після секції коду є різними даними. Відповідно, у секцій даних доступ є або тільки на читання, або на читання/запис. Секції даних,

зазвичай, мають доступ на читання (константи), читання/запис (змінні), а секції коду – на читання і виконання.

Проектовані файли (Mapped Files). Їх позначено синім кольором. Проектування файлів – це зручний спосіб оброблення даних. Передбачають, що дані якби перебувають у пам'яті, хоча, насправді, вони розташовані на диску. Програмістові не потрібно писати явний код завантаження файла в пам'ять, економиться час на переміщенні блоків пам'яті. Усе це робить ОС автоматично, а з погляду програми це звичайна пам'ять.

Як і віртуальна пам'ять, проектовані файли дозволяють резервувати регіон адресного простору і передавати йому фізичну пам'ять. Відмінність між цими механізмами полягає в тому, що в останньому випадку фізична пам'ять не виділяється зі сторінкового файла, а береться з файла, що вже перебуває на диску. Як тільки файл спроектовано в пам'ять, до нього можна звертатися так, ніби він повністю в неї завантажений.

Приватна (private) пам'ять та пам'ять, яка розподіляється (shareable). Їх позначено жовтим і блакитним кольором, відповідно. Велику частину із програми займають звичайні дані, які індивідуальні для програми користувача. Це так звані приватні (private) дані. Ці дані використовуються тільки цією програмою і не мають відображення в інших програмах, тобто це найтиповіша пам'ять програми.

Окрім приватних даних, в адресному просторі є і так звані shareable (які розподіляються) дані. Це спеціальним чином оформлені секції DLL-файлів, які спільно використовуються декількома програмами.

Утрачена (unusable) і вільна (free) пам'ять. Їх позначено сірим і білим кольором, відповідно. Перелічені раніше основні регіони – усе, що можна використовувати. Уся інша пам'ять є недоступною. Будь-яке звернення до неї призводить до порушення доступу (Access Violation) – це окремий випадок неправильного доступу пам'яті. Але навіть недоступну пам'ять розподілено на дві категорії. Найочевидніше – вільна пам'ять. Її більшість.

Окрім вільної пам'яті, у недоступній пам'яті є пам'ять, яку не може бути використано із-за фрагментації. Пам'ять виділяється блоками по 64 КБ, але гранулярність блока пам'яті – це розмір сторінки, тобто 4 КБ. Це означає, що якщо треба виділити 2 КБ, то ОС виділить 4 КБ, а 60 КБ виявляться, навпаки, недоступними – поки не звільняться ці 2 КБ пам'яті. Отже, весь блок 64 КБ не стане знов повністю доступний для використання.

Системна купа (heap). Її позначено червоним кольором. Купа – це назва структури даних, за допомогою якої реалізовано динамічний розподіл

пам'яті додатка. Саме купа (через *Диспетчер пам'яті*) є стандартом для роботи з динамічною пам'яттю. Будь-яка купа є динамічною пам'яттю, але не будь-яка динамічна пам'ять – купою.

Також "купою" називають не тільки сам механізм управління пам'яттю, але й регіон адресного простору, виділений для потреб купи. Спочатку цей регіон є малим або зовсім відсутнім. У міру роботи програми (і виділення в ній пам'яті) спеціальний диспетчер, керівник купами *(Диспетчер пам'яті),* буде розширюватиме цей регіон за потреби або створює нові регіони. А під час звільнення блоків пам'яті в купі Диспетчер пам'яті буде повертати системі відповідні сторінки фізичної пам'яті (в міру можливості).

Таким чином, пам'ять для додатків розподілено за допомогою стандартних функцій бібліотек реального часу C – malloc, HeapAlloc, HeapFree і LocalAlloc і відображається у VMMap червоним кольором.

Керована купа (Managed Heap). Їх позначено зеленим кольором. Керована купа є видом приватної пам'яті процесу, яка виділяється і використовується в .NET "складальником сміття" для непотрібних більше даних або пошкоджених даних.

Стек (Stack). Він позначений помаранчевим кольором. Стек – це структура даних, у якій доступ до елементів організовано за принципом LIFO (last in – first out, "останнім прийшов – першим вийшов"). Операцію уміщення елемента у стек називають "уштовхуванням" (push), а зворотна до неї – "виштовхуванням" (pop). Обидві операції можливі тільки щодо верхівки стека (рис. 6.14).

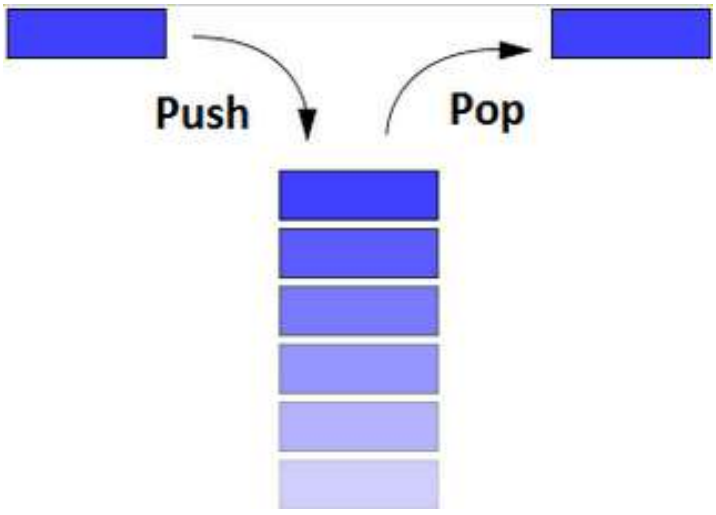


Рис. 6.14. **Схема роботи стека**

В адресному просторі будь-якого процесу є як мінімум один дуже спеціальний регіон пам'яті, так званий "стек потоку". Усякий раз, коли у процесі створюється потік, система резервує регіон адресного простору для стека потоку (у кожного потоку свій стек) і передає цьому регіону якийсь об'єм фізичної пам'яті. За замовчуванням система резервує 1 МБ адресного простору і передає йому всього дві сторінки пам'яті. До речі, стек уміє тільки розширюватися. Одного разу збільшивши свій розмір, він більше не зменшується.

VMMap дозволяє регулярно оновлювати карту розподілу пам'яті, що дозволяє скласти типові карти пам'яті для додатків, які коректно функціонують, для того щоб надалі для полегшення діагностики можна було легко порівняти поточні карти з попередніми та виявити відмінності. Окрім цього, VMMap уміє працювати зі сценаріями й дозволяє експортувати дані у формат .mmp (підтримується тільки цією утилітою). Ще одна дуже корисна можливість VMMap – опція Empty Working Set ("Очистити робочу множину") у меню Refresh ("Відновити"): за допомогою цієї опції можна боротися з надмірним використанням ресурсів пам'яті з боку "ненажерливих" додатків без завершення їхньої роботи або перезавантаження системи. Утім, користуватися цією опцією слід з обережністю і лише в тому разі, якщо відомо напевно, як поведеться після очищення пам'яті той або той процес.

RAMMap v.1.32c (by Mark Russinovich and Bryce). утиліта, яка дозволяє проаналізувати використання фізичної пам'яті у Windows, причому має наочний і зрозумілий інтерфейс із вкладками та передбачає декілька різних режимів подання інформації (рис. 6.15).

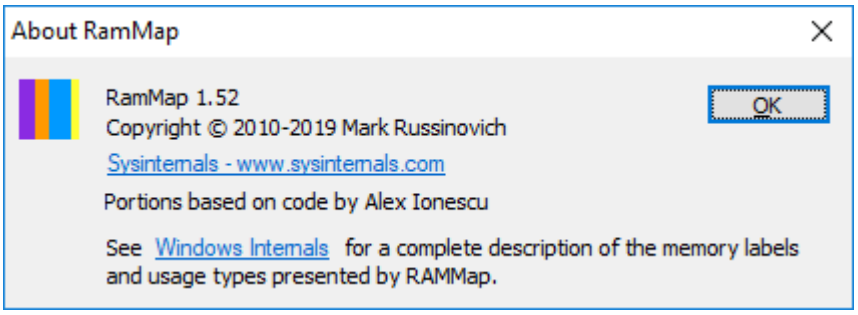


Рис. 6.15. **Діалогове вікно про програму RAMMap**

За допомогою RAMMap можна швидко оцінити об'єми кешованих даних, що зберігають в оперативній пам'яті, дізнатися, скільки пам'яті використовують окремі додатки, драйвери ядра й обладнання, а також дістати

вичерпні відповіді на інші специфічні питання. Для зручності роботи передбачено можливість збереження та завантаження миттєвих знімків поточного стану оперативної пам'яті. RAMMap розповсюджується як у вигляді окремої програми, так і у складі пакета Sysinternals Suite (рис. 6.16).

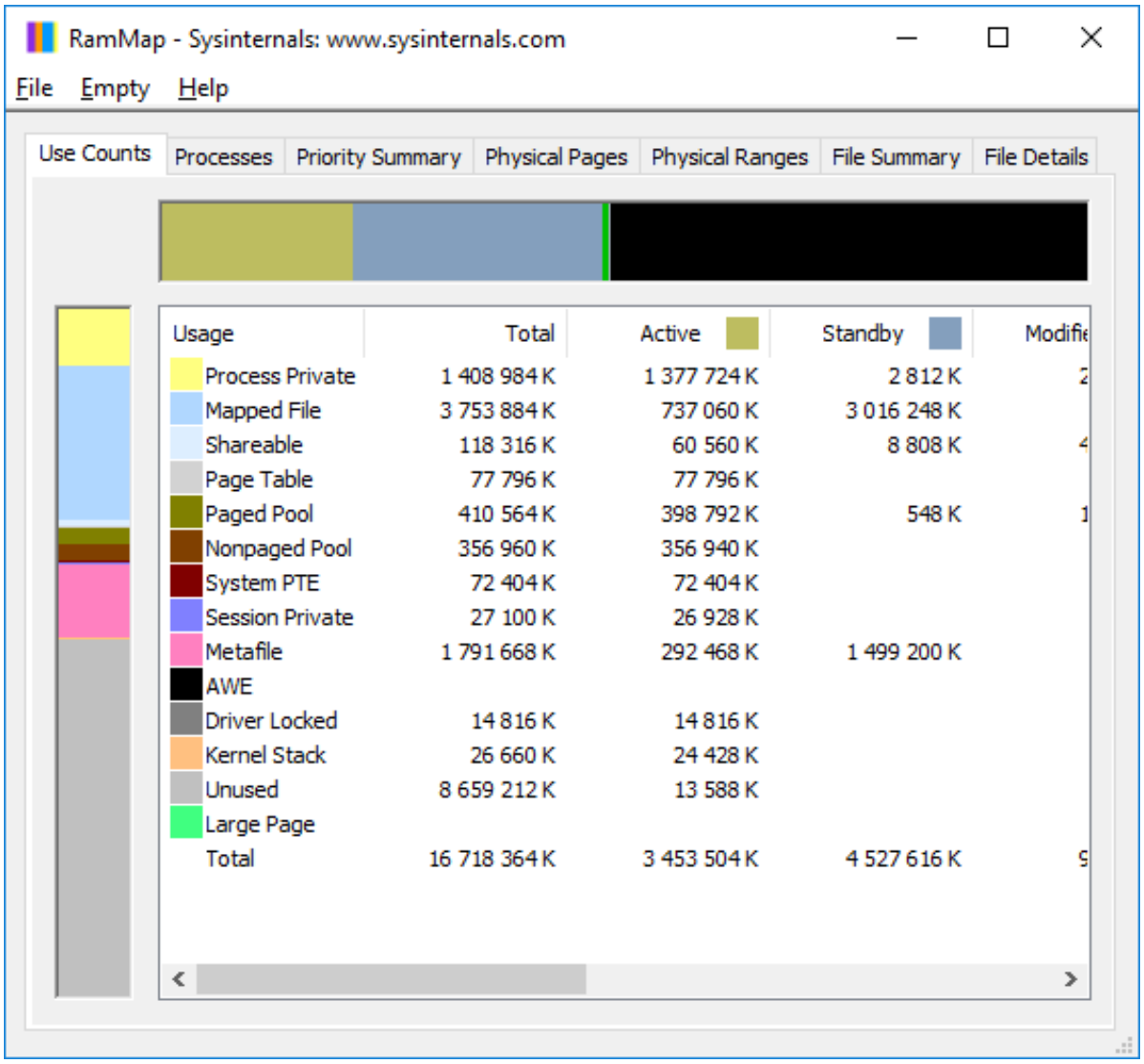


Рис. 6.16. **Головне вікно програми RAMMap**

Wintest. Цей програмний продукт створено на основі трьох демонстраційних програм, розроблених Джеффрі Ріхтером [14]. Тому, програма містить три основні пункти меню:

1. VMStatistica – відповідає програмі VMStat.
2. SysInfo – відповідає програмі SysInfo.
3. VMMap – відповідає програмі VMMap.

1. **VMStat.** Ця програма виводить на екран вікно з результатами виклику функції *GlobalMemoryStatus* Інформація у вікні оновлюється кожну секунду, оскільки VMStat повністю придатна для моніторингу пам'яті в системі. Вікно цієї програми після запуску в Windows показано на рис. 6. 17.

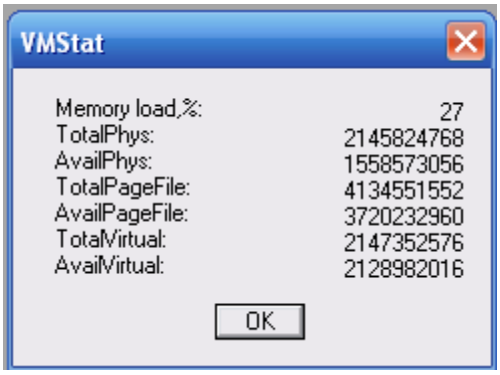


Рис. 6.17. **Діалогове вікно про програму VMStat**

Елемент *dwMemoryLoad* (що показано як Memory Load) дозволяє оцінити, на скільки зайнята підсистема управління пам'яттю. Це число може бути будь-яким в діапазоні від 0 до 100 %.

Елемент *dwTotalPhys* (що показано як TotalPhys) відображає загальний об'єм фізичної (оперативної) пам'яті в байтах. На цій машині його значення становить 2 145 824 768, що на 1 658 880 байтів менш за 2 ГБ. Причина, за якою *GlobalMemoryStatus* не повідомляє про повних 2 ГБ, полягає в тому, що система під час завантаження резервує невелику ділянку оперативної пам'яті, недоступну навіть ядру. Ця ділянка ніколи не скидається на диск. А елемент *dwAvailPhys* (що показано як AvailPhys) дає об'єм байтів вільної фізичної пам'яті.

Елемент *dwTotalPageFile* (що показано як TotalPageFile) повідомляє максимальний об'єм байтів, яка може міститися у сторінковому файлі на жорсткому диску. Хоча VMStat показує, що поточний розмір сторінкового файла становить 4 134 551 552 байти, система може варіювати його на свій розсуд Елемент *dwAvailPageFile* (що показано як AvailPageFile) підказує, що в цей момент 3 720 232 960 байтів у сторінковому файлі вільно і може бути передано будь-якому процесу.

Елемент *dwTotalVirtual* (що показано як TotalVirtual) відображає загальний об'єм байтів, відведених під закритий адресний простір процесу. Значення 2 147 352 576 якраз на 128 КБ менше за 2 ГБ. Два розділи недоступного адресного простору – від 0x00000000 до 0x0000FFFF і від 0x7FFF0000 до 0x7FFFFFFF – якраз і становить цю різницю 128 КБ.

І, нарешті, *dwAvailVirtual* (що показано як AvailVirtual) – це єдиний елемент структури, специфічний для конкретного процесу, який викликає *GlobalMemoryStatus* (решта елементів належить виключно до самої системи й не залежить від того, який саме процес викликає цю функцію). Під час підрахунку значення *dwAvaiWirtual* функція підсумовує розміри всіх вільних регіонів в адресному просторі процесу, що викликається. У цьому разі його значення говорить про те, що в розпорядженні програми VMStat є 2 128 982 016 байтів вільного адресного простору. Віднявши зі значення *dwTotalVirtual* величину *dwAvailVirtual,* знаходять 18 370 560 байтів – такий об'єм пам'яті VMStat зарезервувала у своєму віртуальному адресному просторі. Окремого елемента, який повідомляв би об'єм фізичної пам'яті, використовуваної процесом у цей момент, не передбачено.

2. **SysInfo.** Ця програма вельми проста; вона викликає функцію *GetSystemInfo* та виводить на екран інформацію, повернену у структурі SYSTEM_INFO. Діалогове вікно з результатами виконання програми SysInfo показано на рис. 6.18.

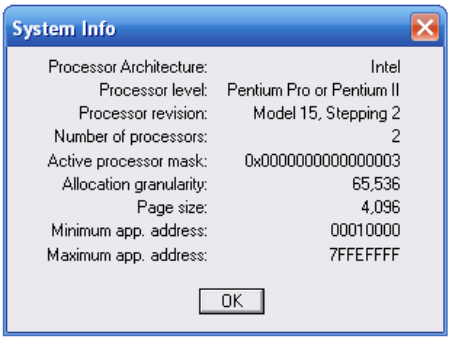


Рис. 6.18. **Діалогове вікно про програму SysInfo**

3. **VMMap.** Ця програма переглядає адресний простір довільного процесу та показує регіони, що містяться в ньому, і блоки, наявні в регіонах. Після запуску VMMap на екрані з'являється наступне вікно.

Кожен елемент у списку – це результат виклику функції *VMQuery.* Основний цикл програми починає роботу з віртуальної адреси NULL і закінчується, коли *VMQuery* повертає FALSE, що вказує на неможливість подальшого перегляду адресного простору процесу. На кожній ітерації циклу викликається функція *ConstructRgnInfoLine.* Вона заповнює символьний буфер інформацією про регіон. Потім ці дані вносять до списку у вигляді табл. 6.1, що підтверджено відповідним скрином (рис. 6.19).

Таблиця 6.1

Список блоків розподілу пам'яті у VMMap

Базові адреси	Типи	Розміри	Блоки	Атрибути захисту	Опис
00000000	Free	65536			
...					

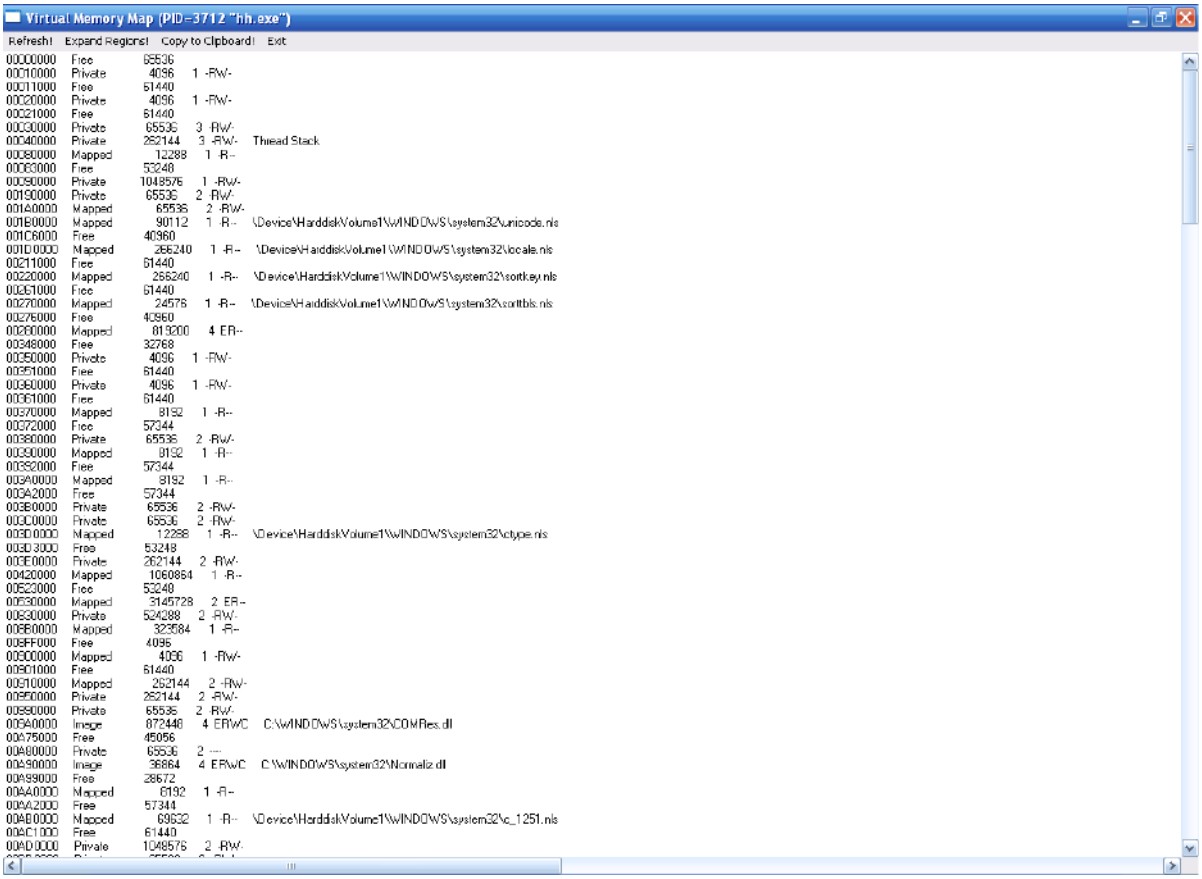


Рис. 6.19. **Результат дослідження розподілу пам'яті у VMMap**

В основний цикл укладено ще один цикл – він дозволяє отримувати інформацію про кожен блок поточного регіону. На кожній ітерації із цього циклу викликається функція *ConstructBlkInfoLine*, що заповнює символьний буфер інформацією про блоки регіону. Ці дані теж додаються до списку. Загалом за допомогою функції *VMQuery* переглядати віртуальний адресний простір процесу дуже легко.

В основі багатьох комп'ютерних архітектур розташовано спеціальний механізм управління пам'яттю, який називають **MMU** (Memory Management

Unit – пристрій управління пам'яттю). На комп'ютерах, оснащених процесором Pentium, механізм MMU є невід'ємною частиною процесора. MMU здійснює перетворення лінійних адрес віртуального адресного простору кожного із процесів, що працюють на комп'ютері, в адреси фізичної ОП, реально встановленої на комп'ютері.

Завдяки MMU здійснюють підтримку багатьох надзвичайно ефективних і зручних технологій. Наприклад, якщо програма звертається до області пам'яті з адресою 0, механізм MMU може переспрямовувати це звернення за адресою 16384. Таким чином, програма буде думати, що здійснює читання (або запис) байта за нульовою адресою, а насправді цей байт розташовано за адресою 16384. Інша програма може також звернутися за адресою 0, проте MMU здійснює перетворення адрес для кожної із програм по-різному. Під час звернення до комірки з нульовою адресою другої програми він переспрямовує це звернення за абсолютно іншою адресою (наприклад, 32768). Інакше кажучи, завдяки MMU кожен процес, що працює в системі, думає, що має власний індивідуальний адресний простір від 0 до 4 ГБ, хоча насправді дані, що належать цим процесам, зберігаються в різних областях одного й того самого фізичного адресного простору.

Якби MMU міг здійснити переспрямування кожного байта ОП за іншою адресою, його таблиці відповідності програмних адрес фізичними були б надмірно великими. Замість цього, MMU розглядає всю оперативну пам'ять як набір невеликих блоків, які називають *сторінками,* і здійснює трансляцію адрес, відповідно до меж цих сторінок. MMU, убудований у процесори типу Pentium, використовує сторінки розміром 4 КБ (інші процесори можуть використовувати інший розмір сторінок). Таким чином, якщо елементу пам'яті з віртуальною адресою 0 відповідає елемент пам'яті з фізичною адресою 16384, то елементу пам'яті з віртуальною адресою 1 завжди буде відповідати елемент пам'яті з фізичною адресою 16385 тощо. Проте елементу пам'яті з віртуальною адресою 4096 може відповідати абсолютно інша, відмінна від 20480 (16384 + 4096) фізична адреса, адже ця комірка належить іншій сторінці пам'яті.

MMU може позначити сторінку віртуальної пам'яті як *відсутню* (absent). Цей механізм лежить в основі організації *віртуальної пам'яті.* MMU повідомляє операційну систему, що відбулося звернення до відсутньої у фізичній оперативній пам'яті сторінки. ОС завантажує відсутню сторінку у фізичну пам'ять і дозволяє програмі знов спробувати звернутися до неї.

Звичайно ж, щоб звільнити місце для завантаження відсутньої сторінки, ОС може записати на диск яку-небудь іншу сторінку віртуальної пам'яті та позначити її як відсутню.

Механізм відсутніх сторінок використовують також для організації *розріджених масивів (sparse arrays).* Уявіть, що ви працюєте з дуже великою матрицею дійсних чисел, для зберігання якої необхідно виділити 10 МБ оперативної пам'яті. Допустімо також, що велика частина елементів матриці дорівнює нулю (або будь-якому іншому константному значенню). Матриці такого роду дуже часто використовують під час вирішення різного виду інженерних завдань.

Невже немає ніякого іншого способу зберігання такої матриці, окрім як виділити пам'ять для того, щоб записати в неї 10 МБ нулів? Windows пропонує таке вирішення проблеми. Усю матрицю розподіляють на сторінки по 4 КБ. Якщо сторінка містить нулі, її позначають як відсутню. Коли програма, що працює з матрицею, звертається до такої сторінки, MMU повідомляє її про те, що сторінка відсутня, і програма розуміє, що всі комірки сторінки насправді дорівнюють нулю. Таким чином, на зберігання сторінок, повністю заповнених нулями, дорогоцінна фізична пам'ять не витрачається.

Загальне завдання для виконання

Виконайте, якщо необхідно, завантаження системного (призначеного для користувача) процесу, указаного у стовпці 2 індивідуального завдання, дослідженню властивостей якого і буде присвячено лабораторну роботу.

До закінчення роботи індивідуальний процес НЕ ВИВАНТАЖУВАТИ з пам'яті

Етап 1. Зробіть аналіз наявної фізичної пам'яті на комп'ютері. Для цього за допомогою *Диспетчера пристроїв* (*Панель управління → Система* або комбінація клавіш Windows + Pause Break) у меню View *(Вигляд)* виберіть команду Resources by Connection *(Ресурси з підключення)* і розкрийте вузол Memory *(Пам'ять)*. Результати занесіть до табл. 6.2.

Таблиця 6.2

Розподіл адрес фізичної пам'яті комп'ютера

Адреси		Розмір блока, КБ	Призначення
початкова	кінцева		
00000000	0009FFFF	640	Системна плата
000A0000	000BFFFF	768	Шина PCI
...	...	...	…
Усього, КБ:		1 408	

Етап 2. Зробіть аналіз використаної фізичної пам'яті та її завантаженість після завантаження операційної системи на комп'ютері. Для цього необхідно використовувати програму FreeRAM (BySoft). Результати, підкріплені скриншотами, занесіть до табл. 6.3.

Таблиця 6.3

Аналіз фізичної пам'яті у FreeRAM (BySoft)

Усього вільної пам'яті, %	Усього пам'яті зайнято, %	Мінімальний рівень пам'яті, потрібний для очищення, МБ

Етап 3. Зробіть аналіз використаної фізичної пам'яті та віртуальної пам'яті на комп'ютері, її завантаженість після завантаження операційної системи. Для цього необхідно використовувати програму Mz RAM Booster. Результати, підкріплені скриншотами, занесіть до табл. 6.4.

Таблиця 6.4

Аналіз пам'яті у Mz RAM Booster

Параметри	Значення параметрів
1	2
Інформація про фізичну пам'ять	
Total Memory, MB	
Used Memory, MB	
Free Memory, MB	

Закінчення табл. 6.4

1	2
Інформація про віртуальну пам'ять	
Total Memory, MB	
Free Memory, MB	
Загальна інформація	
Total Mb Recovered, MB	
Total Optimizations	

Етап 4. Зробіть аналіз фізичної пам'яті за допомогою монітора ресурсів для комп'ютера. Для цього необхідно використати програму Resmon.exe, укладку *Пам'ять* Результати фізичного розподілу пам'яті процесів занесіть до табл. 6.5. Надайте у звіті гістограми розподілу фізичної пам'яті.

Таблиця 6.5

Аналіз фізичної пам'яті в Resmon.exe

Параметри, МБ	Значення параметрів
Зарезервовано апаратно	
Використано	
Змінено	
Зарезервовано	
Вільно	

Етап 5. Зробіть аналіз віртуальної пам'яті процесу, заданого у стовпці 2 індивідуального завдання, за допомогою монітора ресурсів для комп'ютера (див. табл. 6.17). Для цього необхідно використати програму Resmon.exe. Результати розподілу віртуальної пам'яті процесів занесіть до табл. 6.6.

Таблиця 6.6

Аналіз віртуальної пам'яті процесу __*(ім'я процесу)*__ в Resmon.exe

Параметри	Значення параметрів
1	2
Образ	
ІД процесу	
Помилок відсутності сторінки в пам'яті, с	

Закінчення табл. 6.6

1	2
Завершено, КБ	
Робочий набір, КБ	
Загальний, КБ	
Приватний, КБ	

Етап 6. Зробіть аналіз загальних параметрів пам'яті операційної системи. Для цього необхідно використовувати програму FreeRAM XP Pro. Результати аналізу (меню *Файл > Звіт про стан пам'яті*) відобразіть у вигляді табл. 6.7.

Таблиця 6.7

Звіт FreeRAM XP Pro Comprehensive Memory Report

Параметри	Значення параметрів
1	2
Basic Memory Information	
Current Amount of RAM Free	
Amount of Installed RAM	
Available Virtual Memory	
Total Virtual Memory	
Available Total Memory	
Total Memory	
Available Virtual Space	
Total Virtual Space	
Advanced Memory Information	
Commit Limit, Bytes	
Current Committed Bytes	
Percent Committed Bytes In Use	
Cache Bytes	
Cache Bytes Past Maximum	
Pool Paged Allocations	
Pool Paged Bytes	
Free System Page Table Entries	
Pool Nonpaged Allocations	
Pool Nonpaged Bytes	
Total Working Set	
Total Working Set Past Maximum	

Закінчення табл. 6.7

1	2
Cache Statistics	
Copy Read Hits, %	
Data Map Hits, %	
MDL Read Hits, %	
Pin Read Hits, %	
Paging File and Virtual Memory Details	
Paging File Bytes in Use	
Paging File Bytes Past Maximum	
Paging File in Use, %	
Paging File in Use % Past Maximum	
Disk Page File Bytes in Use	
Total Disk Page File Bytes	
Size of a Single Page	
Total Virtual Space Bytes	
Virtual Space Bytes Past Maximum	

Етап 7. Зробіть аналіз параметрів процесу, заданого у стовпці 2 індивідуального завдання. Для цього необхідно використовувати програму FreeRAM XP Pro. Результати аналізу (меню *Інструменти → Звіт про використання RAM)* відобразіть у вигляді табл. 6.8.

Таблиця 6.8

Розширений звіт про процеси у FreeRAM XP Pro

Параметри	Значення
Ім'я файла процесу	
Використовуваний об'єм пам'яті	
Шлях до файла	
Ідентифікатор процесу	
Кількість потоків	
Клас пріоритету процесу	
Ідентифікатор батьківського процесу	
Кількість помилок сторінок	
Максимальний робочий об'єм процесу	
Ємність використовуваної частини процесом файла підкачування	
Максимальний об'єм використовуваного файла підкачування	

Етап 8. Зробіть аналіз системної віртуальної пам'яті комп'ютера, використовуючи програму CleanMem, запустивши її графічну оболонку Mini_Monitor.exe. Відобразіть скриншот. Використовуючи контекстне меню, занесіть до табл. 6.9.

Таблиця 6.9

Аналіз системної віртуальної пам'яті у CleanMem

Параметри	Значення	Опис
Current Size		
Peak Size		
Page Fault Count		
Minimum Working Set		
Maximum Working Set		

Етап 9. Зробіть аналіз основних параметром віртуальної пам'яті процесу, заданого у стовпці 2 індивідуального завдання, за допомогою програми CleanMem, запустивши її графічну оболонку Mini_Monitor.exe. Використовуючи контекстне меню, занесіть дані до табл. 6.10.

Таблиця 6.10

Аналіз віртуальної пам'яті процесу __*(ім'я процесу)*__ у CleanMem

Параметри	Значення	Опис
1	2	3
Caption		
Command Line		
Computer		
Creation Date		
Handle		
Handle Count		
Kernel Mode Time		
Max Working Size		
Min Working Size		
Name		
Parent Process		
Peak Page File Usage		
Peak Virtual Size		

Закінчення табл. 6.10

1	2	3
Peak Working Set		
Priority		
Process ID		
Thread Count		
Virtual Size		

Етап 10. Виконайте спостереження, оптимізацію і тестування пам'яті комп'ютера за допомогою програми RAM Saver Pro. Результати, підкріплені скриншотами, занесіть до табл. 6.11.

Таблиця 6.11

Дослідження пам'яті за допомогою програми RAM Saver Pro

Параметри	Значення параметрів	
	до оптимізації	після оптимізації
Усього RAM, MB		
Вільно RAM, MB		
Усього pagefile, GB		
Вільно pagefile, GB		

Етап 11. Виконайте тестування параметрів фізичної пам'яті комп'ютера (стовпці 3, 4 індивідуального завдання). Результати дослідження швидкостей операцій занесіть до табл. 6.12 і побудуйте графік залежності від кількості потоків. Зробіть висновок.

Таблиця 6.12

Тестування пам'яті за допомогою програми RAM Saver Pro

Операції	Кількість потоків									
	1	2	3	4	5	6	7	8	9	10

Етап 12. Зробіть евристичне дослідження віртуального адресного простору процесу, заданого у стовпці 2 індивідуального завдання, використовуючи програму VMMap.exe. Для цього після запуску програми

в діалоговому вікні зробіть вибір необхідного процесу. Після чого визначте статистичну й адресну інформацію про розподіл віртуальних адрес, характеристику системних просторів процесу, купи (heap), стеків тощо.

Як результати подайте розподіл усього адресного простору у вигляді скриншота, як показано далі (рис. 6.20), а також у вигляді табл. 6.13 розподілу адрес.

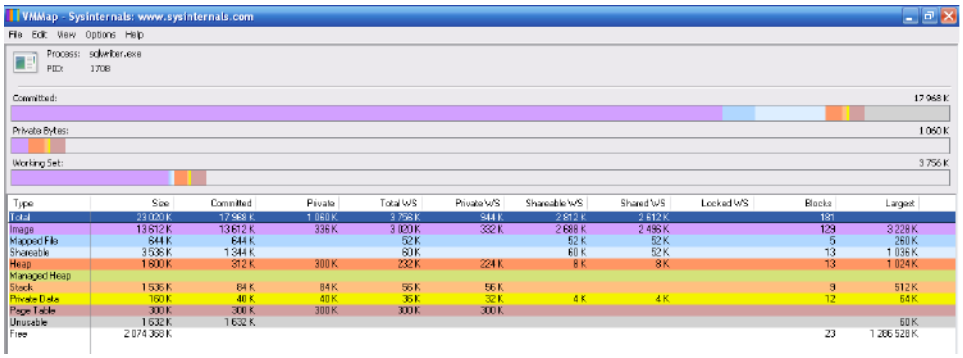


Рис. 6.20. **Розподіл адресного простору процесу у VMMap**

Таблиця 6.13

Розподіл віртуального адресного простору процесу
___(ім'я процесу)___

Адреси	Типи блоків	Розміри	Захист	Опис
01000000	Image	..	Execute/Read	D:\...\..exe
..				
7FFFFFFF				

Етап 13. Зробіть аналіз розміщення образу файла, заданого у стовпці 2 індивідуального завдання, у фізичному і віртуальному адресних просторах. Для цього необхідно використовувати програму RAMMap. Результати занесіть до табл. 6.14.

Таблиця 6.14

Аналіз розміщення образу файла у віртуальній пам'яті

Physical address		List	Use	Priority	Image	Offset	File Name	Process	Virtual address	
початковий	кінцевий								початковий	кінцевий

Етап 14. Проведіть дослідження пам'яті, використовуючи програму WinTest.exe за допомогою трьох підпрограм: VMStat, SysInfo, VMMap. В останній програмі після запуску програми в діалоговому вікні зробіть вибір необхідного процесу. Після чого визначте статистичну й адресну інформацію про розподіл віртуальних адрес, характеристику системних просторів процесу. Як результати подайте відомості про фізичну і віртуальну пам'ять у вигляді табл. 6.15, розподіл усього адресного простору процесу – у вигляді табл. 6.16.

Таблиця 6.15

Характеристика пам'яті та системної інформації

Параметри	Значення	Параметри	Значення
VMStat		SysInfo	
Memory load,%		Processor Architecture	
TotalPhys		Processor level	
AvailPhys		Processor revision	
TotalPageFile		Number of processors	
AvailPageFile		Active processor mask	
TotalVirtual		Allocation granularity	
AvailVirtual		Page size	

Таблиця 6.16

Розподіл адресного простору процесу __*(ім'я процесу)*___

Базові адреси	Типи	Розміри	Блоки	Атрибути захисту	Опис
00000000	Free	65536			
...					

Після закінчення роботи варто очистити від прописаних параметрів автозапуску програм у гілки системного реєстру HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Додаткове завдання

Створіть додаток, у якому слід передбачити наявність мінімум трьох пунктів головного меню. Один пункт меню управляє операціями з вікном (згорнути, розгорнути, розмір, перемістити, закрити). Другий відповідає виконанню п. 2 індивідуального завдання, відповідно до варіанта. Третій пункт меню відображає інформацію про автора-розробника програми зі вказівкою факультету, номера навчальної групи та прізвища студента. Забезпечте підтримку довільного набору гарячих клавіш для кожного пункту меню (табл. 6.17).

Таблиця 6.17

Індивідуальні завдання для виконання

Варіанти	Об'єкти дослідження	Типи тестів	Типи операцій	Додаткові завдання
1	2	3	4	5
1	notepad.exe	Doubled access	Читання	Створіть додаток, за допомогою якого можна було б візуально досліджувати розподіл віртуальної пам'яті процесів. Як можливі процеси використовуйте програми WINWORD, EXCEL і CALC. Дайте можливість користувачеві право вибору для дослідження необхідного процесу. Як статистичну інформацію слід використовувати відображення логічної адреси областей пам'яті, стани (вільний, готовий до використання або зарезервований), типи доступу, об'єми блоків пам'яті, базові адреси, тип об'єктів
2	colorcpl.exe	Linear access	Запис	Під час запуску додатка створюється буфер заданого користувачем розміру. Буфер призначено для зберігання *N* символів і нульової ознаки кінця. Дайте можливість користувачеві ввести в буфер дані про прізвище, ім'я та по батькові розробника програми, назву факультету, номер групи та курсу. Якщо об'єм буфера буде недостатньо, передбачте запит на його збільшення, а якщо первинний розмір буфера буде більшим, то виконайте його зменшення. Таким чином, буфер має бути заповнено повністю

Продовження табл. 6.17

1	2	3	4	5
3	shrpubw.exe	Doubled access	Копіювання	Побудуйте додаток, у якому під час запуску додатку створюється купа об'ємом 100, 200, 500, 5000 байтів. Задавання об'єму виконувати в окремому вікні. Потім пам'ять має бути перерозподілено і використовуватися як динамічний масив рядків (уміст задайте самостійно, однотипні рядки не допускають). Коли користувач вибирає один із пунктів меню, із купи розподіляється пам'ять для зберігання нового рядка і до масиву додається покажчик на цю пам'ять. Якщо в масиві не залишається місця, має бути запит на розширення масиву. Забезпечте виведення у клієнтській зоні вікна вмісту рядків і значення покажчика. Якщо користувач вибирає інший пункт меню, то звільняється останній розподілений рядок
4	charmap.exe	Linear access	Читання	Створіть додаток, розподіліть сторінки віртуальної пам'яті. У діалоговому режимі забезпечте можливість виділення довільного блоку віртуальній пам'яті для процесу. За наслідками виділення зробіть звіт, який має містити такі дані, як: об'єм (у байтах) зарезервованого регіону; тип фізичної пам'яті, використовуваний групою блоків цього регіону: MEM_FREE, MEM_IMAGE, MEM_MAPPED або MEM PRIVATE; кількість блоків у вказаному регіоні. Передбачте додаткове виділення нових блоків, так само як і видалення вже виділених
5	msiexec.exe	Doubled access	Запис	Створіть додаток, що забезпечує аналіз параметрів і вмісту віртуальної пам'яті. Визначте відсоток поточного використання комп'ютером своїх ресурсів, об'єму фізичної пам'яті та файла підкачування, об'єму вільного місця у файлі підкачування, об'єму вільного місця на диску. Забезпечте можливість блокування/розблокування об'єктів пам'яті. Доведіть, що є функції розблокування, що видаляють тільки дескриптор глобального об'єкта пам'яті, а також є дескриптори, які додатково видаляють із пам'яті розблоковані сторінки

Продовження табл. 6.17

1	2	3	4	5
6	charmap.exe	Linear access	Копіювання	Створіть додаток, розподіліть сторінки віртуальної пам'яті. У діалоговому режимі забезпечте можливість виділення довільного блока віртуальній пам'яті для процесу. За наслідками виділення зробіть звіт, який має містити такі дані, як: об'єм (у байтах) зарезервованого регіону; тип фізичної пам'яті, використовуваний групою блоків цього регіону: MEM_FREE, MEM_IMAGE, MEM_MAPPED або MEM PRIVATE; кількість блоків у вказаному регіоні. Передбачте додаткове виділення нових блоків, так само як і видалення вже виділених
7	magnify.exe	Doubled access	Читання	Створіть додаток, у якому під час запуску виконують резервування віртуальної пам'яті заданого користувачем об'єму. Коли користувач вибирає один із пунктів меню додатка, то виділяються і використовуються 100 КБ віртуальної пам'яті. Значення розміщуються в кожному блоці пам'яті об'ємом 1 КБ. Доступ до всього виділеного блока пам'яті змінюється на доступ "тільки для читання". Відбувається доступ до значення у блоці пам'яті та його відображення у вікні повідомлень. За допомогою іншого пункту меню користувач має зробити спробу змінити значення в пам'яті
8	cleanmgr.exe	Linear access	Запис	Розробіть додаток, що взаємодіє із системним кешем. Під час запуску додатка має створюватися купа, після чого пам'ять перерозподіляється і використовується у вигляді масиву рядків певного розміру. Коли користувач вибирає один пункт меню з купи, розподіляється пам'ять для зберігання нового рядка і до масиву додається покажчик на цю пам'ять. Передбачте відображення у клієнтській області вмісту покажчиків і самих масивів. Якщо розмір купи не достатній, має видаватися користувачеві повідомлення для того, щоб він міг іншим пунктом меню звільнити у зворотному порядку пам'ять, виділену рядкові

Продовження табл. 6.17

1	2	3	4	5
9	mobsync.exe	Doubled access	Копіювання	Побудуйте додаток – аналізатор параметрів віртуального адресного простору процесу, який має здійснювати виведення або у клієнтську зону вікна, або в діалогове вікно таку інформацію, як: архітектуру процесора; розмір сторінки; покажчики на молодшу і старшу адресу пам'яті, доступну для додатка; маску активних процесорів системи; кількість процесорів у системі; тип процесора; мінімальний фактично резервований адресний простір для функції VirtualAlloc(); рівень системи, залежно від архітектури використовуваного процесора; модифікацію процесора
10	mmc.exe	Linear access	Читання	Розробіть додаток – моніторинг пам'яті в системі. Опціями додатку задайте час оновлення даних вікна моніторингу. Вікно моніторингу має містити докладну інформацію і про фізичну, і про віртуальну пам'ять, а також дозволяти обробляти об'єми пам'яті, що перевищують 4 ГБ. До такої інформації має належати: поточний стан використання пам'яті (число 0 ... 100); загальний об'єм оперативної пам'яті; об'єм байтів доступної фізичної пам'яті; загальний об'єм байтів файла підкачування; загальний об'єм байтів, доступних із файла підкачування; загальний об'єм байтів віртуальної пам'яті, доступної поточному процесу; об'єм незарезервованих і невиділених байтів віртуальної пам'яті; об'єм незарезервованих і невиділених байтів віртуальної пам'яті в розширеній частині віртуального адресного простору поточного процесу
11	narrator.exe	Doubled access	Запис	Розробіть додаток, за допомогою якого користувач міг би вводити довільні адреси елементів віртуальної пам'яті та об'єм байтів (КБ), а в робочій області додатка виводився б їхній уміст із розшифруванням значення належності до певного модуля

Продовження табл. 6.17

1	2	3	4	5
12	eudcedit.exe	Linear access	Копіювання	Розробіть додаток, за допомогою якого користувач міг би вибрати довільний процес і, проаналізувавши його віртуальний адресний простір, відобразити вміст тільки тій частині, яка використовується різними системними *.dll. Відображення має показувати кількість блоків, атрибути захисту, належність до dll, тип (free, private, image або maped) і, звичайно ж, базові адреси
13	msinfo32.exe	Doubled access	Читання	Під час первинного запуску додатку резервується буфер пам'яті. Об'єм буфера налаштовується у відповідному діалоговому вікні. Буфер призначено для зберігання *N* символів і нульової ознаки кінця. У буфер автоматично розміщують дані про розробника програми, номера групи та курсу, причому повністю заповнюючи буфер. Передбачте відповідні обмеження. Ці відомості виводяться у клієнтську зону вікна. Під час вибору користувачем одного з пунктів меню (задайте самостійно) блок пам'яті перерозподіляється, щоб звільнити місце ще для *K* символів. Ці додаткові символи записуються в буфер, означають дату народження автора програми та відображаються у клієнтській зоні вікна. Під час вибору іншого пункту меню потрібно постійно очищати вміст клієнтської зони вікна
14	sigverif.exe	Linear access	Запис	Створіть додаток, що робить аналіз параметрів і вмісту віртуальної пам'яті. Визначте відсоток поточного використання комп'ютером своїх ресурсів, об'єму фізичної пам'яті та файла підкачування, об'єму вільного місця у файлі підкачування, об'єму вільного місця на диску. Крім того, використовуючи як управління іншими пунктами меню, дайте можливість користувачеві встановлювати для довільних сторінок пам'яті прапорець GMEM_MOVEABLE

Продовження табл. 6.17

1	2	3	4	5
15	wscript.exe	Doubled access	Копіювання	Створіть додаток, що дозволяє аналізувати інформацію про доступну як фізичну, так і віртуальну пам'ять, причому передбачте можливість перевищення загального об'єму 4 ГБ. Особливу увагу приділіть аналізу параметрів файла підкачування. Дайте можливість користувачеві резервувати/звільняти серед вільного діапазону довільний об'єм пам'яті
16	iexpress.exe	Linear access	Читання	Створіть додаток, у якому відбувається резервування певного об'єму пам'яті. Досліджуйте типи помилок, що виникають, залежно від типу доступу до пам'яті. Надайте список усіх можливих помилок типу доступу. Під час вибору конкретної помилки виконують її моделювання. Забезпечте довідність і наочність, що підтверджують наявність цієї помилки
17	eventvwr.exe	Doubled access	Запис	Побудуйте додаток, що дозволяє зберігати рядок у блоці пам'яті глобальній купі. Уміст рядка має містити назви країн світу. Під час первинного запуску додатка має виконуватися розподіл цього рядка у 27-байтному буфері. Після чого цей рядок має відобразитися у клієнтській зоні вікна. Під час вибору пункту меню блок пам'яті перерозподіляється для звільнення місця ще для 26 символів. Ці додаткові символи записуються в буфер у вигляді назв країни задом наперед і відображаються у клієнтській зоні вікна, де також указано значення прапорів глобальної купи: GHND; GMEM_FIXED; GMEM_MOVEABLE; GMEM_ZEROINIT; GPTR
18	mstsc.exe	Linear access	Копіювання	Розробіть додаток, що здійснює вибір довільного процесу як призначеного для користувача, так і системного, і відображення у клієнтській області вікна розподіл адресного простору цього процесу

Продовження табл. 6.17

1	2	3	4	5
19	syskey.exe	Doubled access	Читання	Після запуску додатка створюється три підпункти меню. За допомогою першого – запущеному процесу виділяється сторінка віртуальної пам'яті, заповнена двійковими одиницями. Під час вибору другого – розкривається підменю, у якому користувач міг би вручну встановлювати типи доступу для однієї сторінки віртуальної пам'яті. Під час вибору третього – звільняється виділена сторінка віртуальної пам'яті. Передбачте можливість задавання тільки одного типу доступу
20	verifier.exe	Linear access	Запис	Розробіть додаток, за допомогою якого досліджують механізм віртуальної пам'яті для управління масивом структур. Спочатку для масиву не резервують ніякого регіону, і весь адресний простір, призначений для нього вільний, що й відображено на карті пам'яті. Карту пам'яті має бути виділено: жовтим кольором – вільні ділянки; синім – зарезервовані; червоним – виділені для використання блоки. За допомогою діалогового вікна здійсніть резервування області деякого об'єму (< 100 КБ). Після цього вводять індекс. Водночас за адресою, де має розташовуватися вказаний елемент масиву, передається фізична пам'ять. Далі карта пам'яті знов перемальовувалася і вже відображає стан області, зарезервовану під весь масив. Будь-який елемент масиву, позначений як зайнятий, можна звільнити клацанням кнопки Clear
21	dxdiag.exe	Doubled access	Копіювання	Створіть додаток, розподіліть сторінки віртуальної пам'яті. У діалоговому режимі забезпечте можливість виділення довільного блока віртуальній пам'яті для процесу. За наслідками виділення зробіть звіт, який має містити такі дані, як: ідентифікатор базової адреси регіону віртуального адресного простору; атрибути захисту, надані регіону під час його резервування. Передбачте додаткове виділення нових блоків, так само як і видалення вже виділених

Продовження табл. 6.17

1	2	3	4	5
22	osk.exe	Linear access	Читання	Розробіть додаток, що резервує віртуальну пам'ять об'ємом 1,5 МБ під час його запуску. Під час вибору другого пункту меню завдання виконуються виділення для використання блока 10 КБ віртуальної пам'яті. Водночас змістіть у кожен 1 КБ блока значення від 1 до 10, відповідно. Виведіть у клієнтську область результати. Дайте можливість користувачеві встановлювати доступ за кожним із 1 КБ або тільки для читання, або читання і запис, або заборонити доступ. Дайте можливість користувачеві самостійно задати нове значення будь-якого 1 КБ з виділеного блока. Зробіть аналіз помилок захисту пам'яті, що виникають, і відобразіть їх у звіті
23	cmstp.exe	Doubled access	Запис	Розробіть програму, яка намагається заповнити сторінку за сторінкою у клієнтській або системній областях віртуального адресного простору значенням, заданим користувачем. Користувач в опціях системи повинен мати можливість як вибору областей, завдання діапазонів адрес, так і значення байта заповнювача. За спроби невдалого заповнення зробіть аналіз, наприклад, відсутність реальної сторінки за вказаною адресою або може бути заборонено до неї доступ. Результати має бути виведено у клієнтській зоні вікна та збережено на диску
24	dialer.exe	Linear access	Копіювання	Розробіть додаток, що здійснює вибір довільного процесу як призначеного для користувача, так і системного. У клієнтській зоні вікна зробіть відображення розподілу всього віртуального адресного простору з обов'язковою вказівкою таких параметрів, як: базова адреса області; тип області; розмір області в байтах; кількість блоків у зарезервованій області; атрибути захисту області пам'яті; короткий опис умісту поточної області (повний шлях до файла)

Продовження табл. 6.17

1	2	3	4	5
25	calc.exe	Doubled access	Читання	Після запуску додатка створюється три підпункти меню. За допомогою першого – запущеному процесу виділяється сторінка віртуальної пам'яті, заповнена двійковими одиницями. Під час вибору другого – розкривається підменю, у якому користувач міг би вручну встановлювати типи доступу для однієї сторінки віртуальної пам'яті. Під час вибору третього – звільняється виділена сторінка віртуальної пам'яті. Передбачте можливість задавання тільки одного типу доступу
26	credwiz.exe	Linear access	Запис	Створіть додаток, у якому відбувається резервування певного об'єму пам'яті та надання йому прав певного доступу. Дослідіть типи помилок, що виникають, залежно від типу доступу до пам'яті. Подайте список усіх можливих помилок типу доступу. Під час вибору конкретної помилки відбувається її моделювання. Забезпечте довідність і наочність, що підтверджують наявність цієї помилки
27	psr.exe	Doubled access	Копіювання	Створіть додаток, який за вибором користувача відповідних пунктів меню відображав у структурованому вигляді локальну таблицю дескрипторів пам'яті процесу. Відображення має вміщувати докладний коментар призначення полів таблиць. Під час вибору одного з пунктів меню виконайте додаткове виділення блоків віртуальної пам'яті заданих користувачем об'ємів
28	mspaint.exe	Linear access	Читання	Побудуйте додаток, що моделює застосування функції GlobalDiscard(). Для цього під час ініціалізації додатка виконайте розподіл пам'яті на декілька блоків однакового розміру, у кожен із яких розмістивши назву факультетів університету. Виведіть у діалогове вікно повний уміст вибраного користувачем блока. Після цього застосуйте у вказаному блоці GlobalDiscard(). Постійно здійснюйте виведення дескрипторів глобального об'єкта пам'яті, поверненого функцією GloballAlloc(). Після чого виконайте повторний розподіл пам'яті функцією GloballReAlloc()

Закінчення табл. 6.17

1	2	3	4	5
29	write.exe	Doubled access	Запис	Побудуйте додаток, у якому меню має містити два підпункти. Під час вибору першого – розкривається підменю, у якому користувач міг би вручну встановлювати для довільно взятої зі списку сторінки віртуальної пам'яті процесу один із таких станів: вільна, заповнена нулями, правильна (використовується активним процесом), змінена, запасна, погана. Під час вибору другого – звільняється виділена сторінка віртуальної пам'яті. Передбачте можливість задавання довільних комбінацій типів доступу, а також вилучіть взаємовиключні типи
30	MdSched.exe	Linear access	Копіювання	Розробіть додаток, що дозволяє у другому пункті свого меню вибирати системний процес, наприклад, lsass.exe або csrss.exe. Проведіть для нього дослідження віртуального адресного простору адрес, що відводиться процесу. Виведення статистичної інформації виведіть у додаткове вікно. До такої статистичної інформації про адресний простір мають належати логічні та фізичні адреси областей пам'яті процесу, їхній тип, статус, розмір, інформація стосується використання системних файлів – динамічних бібліотек

Контрольні запитання

1. Поясніть архітектуру пам'яті в операційній системі Windows.

2. Назвіть призначення і місце в архітектурі Windows глобальної пам'яті.

3. Яким чином здійснюється взаємодія між оперативною пам'яттю і файлом підкачування?

4. Поясніть способи реєстрації зміни станів сторінок віртуальної пам'яті та механізми їхнього переміщення на диск і назад.

5. Як розподілено системну віртуальну пам'ять?

6. Дайте визначення поняттю купи. Які функції використовують для роботи з нею?

Розділ 3. Файлова система

Лабораторна робота 7
Дослідження виконуваного файла Windows

Мета роботи: вивчення структури виконуваних типів файлів в операційній системі Windows, здійснення аналізу структури файлового формату Portable Executable (PE), ознайомлення з розміщенням ресурсів у виконуваних файлах Windows, використовуваними програмними засобами сторонніх розробників, що дозволяють уносити корективи до параметрів різних типів ресурсів виконуваних файлів.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно вивчити структуру PE – рідного файлового формату Windows, порядок внутрішньої будови виконуваних файлів Windows. Додаткову інформацію під час підготовки до роботи можна отримати в [17; 19].

Теоретичні відомості

PE означає Portable Executable. Це "рідний" файловий формат Win32. Його специфікації походять від Unix Coff (common object file format). Portable executable означає, що файловий формат універсальний для платформи win32: завантажувач PE будь-якої win32-платформи розпізнає і використовує це файловий формат, навіть коли Windows запускається на не PC CPU-платформі, хоча це не означає, що вашим PE можна буде портувати на інші CPU-платформи без змін. Кожен win32-виконавчий файл (окрім VXD і 16-бітових DLL) використовує PE-формат. Навіть драйвери ядра NT використовують PE-формат. От чому знання цього формату допоможе краще пізнати внутрішню структуру Windows.

Важливим поняттям, про яке необхідно знати, є RVA (Relative Virtual Address – відносна віртуальна адреса). Багато полів у PE-файлах задають саме за допомогою їхніх RVA. RVA – це просто зсув цього елемента

щодо адреси, із якої починається відображення файла в пам'яті. Хай, наприклад, завантажувач Windows відобразив PE-файл у пам'ять, починаючи з адреси 400000h у віртуальному адресному просторі. Якщо якась таблиця у відображенні починається з адреси 401464h, то RVA цієї таблиці 1464h:

Virtual Address – Image Base = Relative Virtual Address
401464h – 400 000h = 1464h

Щоб перевести RVA в покажчик пам'яті, необхідно додати RVA до базової адреси, починаючи з якої було завантажено модуль. Термін "базова адреса" становить ще одне важливе поняття, про яке слід пам'ятати. Базова адреса – це адреса, із якої починається відображений у пам'яті EXE-файл або DLL. Для зручності Windows використовують базову адресу модуля як дескриптор образу модуля (HINSTANCE – instance handle).

Для дослідження структури виконуваних файлів слугують такі програмні засоби, як:

1. *PE Explorer v1.99* (by Heaventools).
2. *PEview version 0.9.9.0* (by Wayne J. Radburn).
3. *PEiD v0.95* (by snaker, Qwerton, Jibz & xineohP).

PE Explorer v1.99. PE Explorer є найбільш функціонально насиченою програмою для перевірки внутрішньої роботи виконуваних файлів EXE, DLL, SYS, DRV, CPL, OCX, BPL, DPL, SCR та ін., для яких у користувача немає початкового коду.

PE Explorer дає можливість відкривати, переглядати та редагувати PE-файли, виконувати статичний аналіз, виявити багато інформації про функції виконуваного файла і зібрати якомога більше інформації про виконуваний файл, наскільки це можливо без його виконання.

PE Explorer залишає користувачеві мінімум роботи для аналізу фрагментів програмного коду. Як тільки вибрано аналізований файл PE Explorer перевіряє його і відображає у вигляді резюме інформацію про PE-заголовки та всі ресурси, що містяться у файлі PE. Звідси інструмент дозволяє досліджувати конкретні елементи у виконуваному файлі (рис. 7.1).

Рис. 7.1. **Головне вікно програми PE Explorer**

Окрім того, PE Explorer містить PE Header Viewer, Exported/Imported API Function Viewer, API Function Syntax Lookup, Resource Viewer/Editor, Dependency Scanner & Disassembler.

PE Explorer призначено для використання в таких різних сценаріях, як розроблення програмного забезпечення, зворотний інжиніринг, зворотний аналіз безпеки та двійковий аудит процесів.

PEview version 0.9.9.0. Розроблено програму Вейном Дж. Редберном, вона розповсюджується абсолютно безкоштовно як спеціальний додаток до програм аналізаторів PE виконуваних файлів. Написано програму на асемблері. Утиліта PEview забезпечує простий і швидкий спосіб перегляду структури та вмісту файлів, записаних у PE і COFF-форматах. Дозволяє переглядати файли типу EXE, DLL, SYS, OCX, CPL, SCR, OBJ, LIB, DBG та ін. (рис. 7.2).

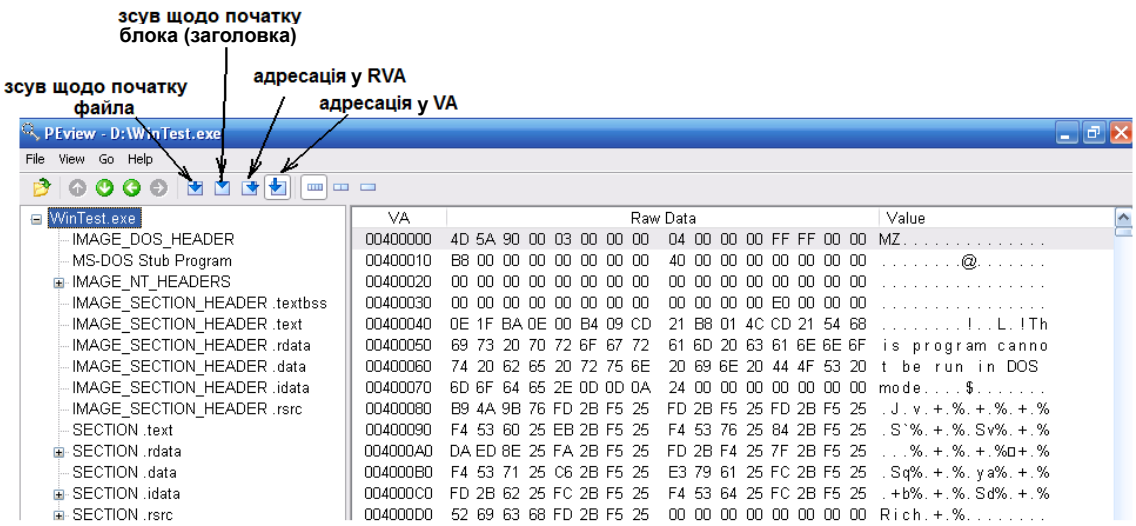


Рис. 7.2. **Головне вікно програми PEview**

PEiD v0.95 (PE iDentifier) – це інструмент для дослідження PE-файлів, що дозволяє дізнатися мову програмування, використану під час написання програми, назву пакувальника, за допомогою якого цю програму стиснуто, або криптора, яким програму зашифровано (рис. 7.3).

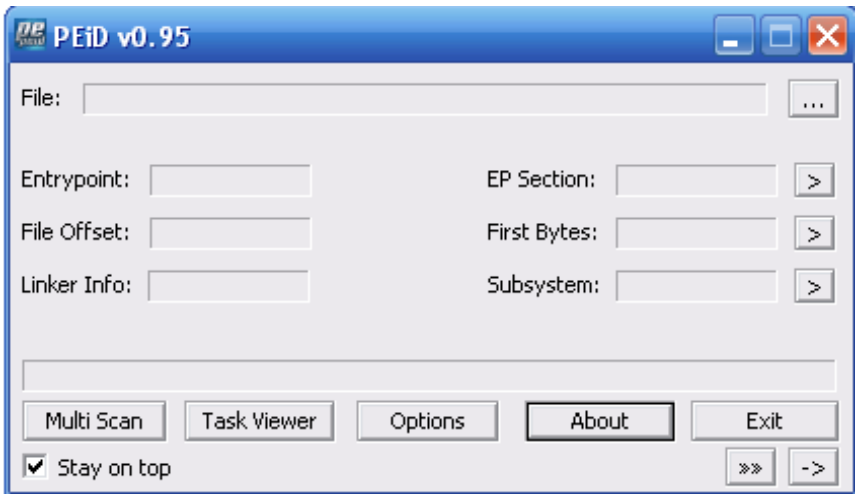


Рис. 7.3. **Головне вікно програми PEiD**

Аналіз програмою PEiD здійснюється по внутрішній і зовнішній базі сигнатур, поточна версія 0.95 може визначати **672** різних сигнатур у PE-файлах. У програмі PEiD є декілька рівнів сканування, можна обробляти цілі каталоги.

Розширити функціонал PEiD можна за рахунок використання плагінів, які можна скачати з офіційної сторінки PEiD або знайти у сторонніх розробників. Опис можливостей і функціонала програми можна знайти у файлі readme.txt.

У лівому нижньому кутку checkbox *Stay on top*. Якщо його позначено, робоче вікно програми PEID завжди буде зверх інших вікон, якщо знято, то поведінка вікна буде звичайною.

Зверху вікна позначка *File*: текстове віконце й невелика кнопочка. Усе це слугує для вибору та відображення файла, із яким буде працювати програма PEiD. Програма підтримує метод Drag-and-Drop. Після завантаження файла програма починає його аналіз.

У разі, якщо PEiD нічого не вдалося знайти, вона виведе *Nothing found*. Якщо завантажений файл є не PE-файлом, програма видає *Not a valid PE file.*

Аналіз PE-файла здійснюється за допомогою сигнатур. Про сигнатури написано у файлі external.txt. Самі зовнішні сигнатури перебувають у файлі userdb.txt.

Модифікацію секції ресурсів виконуваного файла може бути здійснено за допомогою додаткових програмних засобів.

Resource Hacker. Ресурси у яких-небудь виконуваних файлах (зокрема 32bit exe's, dll's, ocx's і cpl's) Windows можна розглядати, вибираючи File → Open з меню Resource Hacker. Повний список ресурсів файла буде показано в деревоподібній структурі. Дерево ресурсу може бути повністю розширене або зруйноване, вибираючи Дерево View → Expand або Дерево View → Collapse, відповідно, з меню (рис. 7.4).

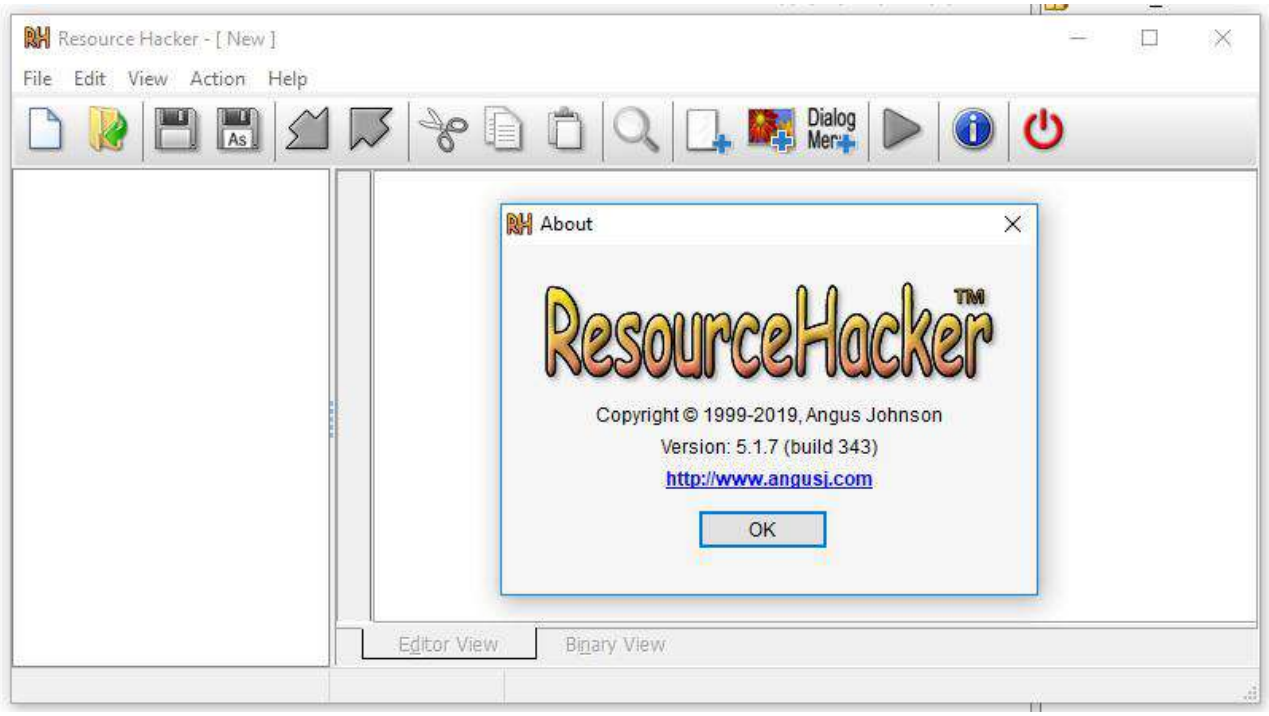


Рис. 7.4. **Головне вікно програми PEview**

Спеціальним елементом ресурсу є тип ресурсу, його ім'я та ідентифікатор мови.

Ресурси групуються в типи ресурсів. Там є цілий ряд заздалегідь визначених типів (іконки, курсори, точкові рисунки, діалоги, меню, rcdata, тощо) ресурсу, але програміст може також визначити інші типи ресурсу.

Елементи ресурсу запам'ятали в межах їхніх відповідних типів ресурсу і мають "Ім'я ресурсу", який унікальний у межах того типу. Це "ім'я"

може зазвичай бути або цілим значенням, або буквено-цифровим. Проте деякі типи (наприклад, таблиці рядків) ресурсу дозволяють тільки цілі значення імені.

Кожен названий ресурс може мати більш ніж один елемент мови, щоб дати можливість програмам управляти багаторазовими мовами. Під кожним ім'ям ресурсу в дереві ресурсу з'явиться як мінімум один елемент "мови ресурсу". Ідентифікатор мови – це ціле значення слова, зроблене з первинної мови і його підмови, визначеної Windows. Якщо елемент ресурсу – це "нейтральна мова", то значення буде нулем.

Multilizer – це могутня за своїми можливостями програма для створення локалізацій додатків, а простіше кажучи, для перекладу меню програм і решти всіх ресурсів іншими мовами (рис. 7.5).

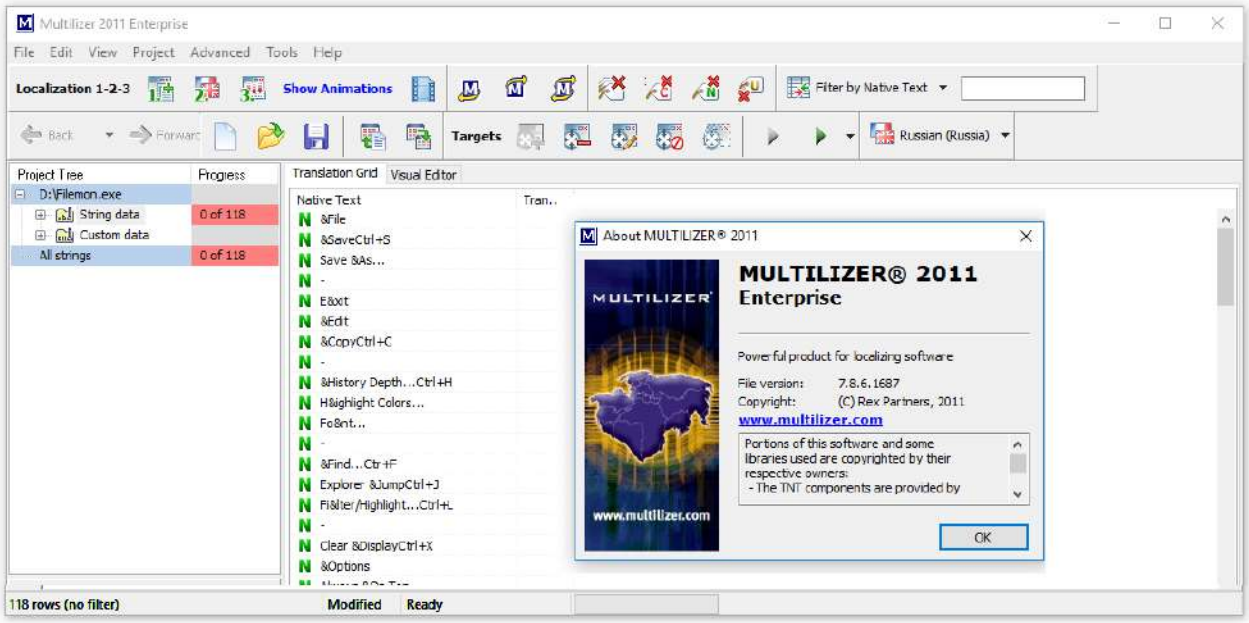


Рис. 7.5. **Головне вікно програми Multilizer**

Multilizer розуміє велику кількість форматів, розрізняє тип мови програмування, на якому написано програму, дозволяючи редагувати програми, призначені для різних операційних систем. Програма підтримує велику кількість опцій, що дозволяють не те що без особливих зусиль, але навіть із комфортом, займатися локалізацією, зокрема є функція імпорту у проект уже готових перекладів.

LikeRusXP – це єдина у своєму роді програма універсальний русифікатор. Перекладає будь-які програми, бібліотеки, а також файли будь-яких

інших форматів і змісту російською мовою з англійської. Користувачеві не треба більше скачувати русифікаторів до програм або перекладати програми вручну за допомогою програм перегляду ресурсів, витрачаючи трафік і свій час. Достатньо тільки вказати, який файл необхідно перекласти, і програма сама автоматично перекладе його.

Програма має вбудований візуальний редактор ресурсів із можливістю перекладу, сканер коду, кінцевий русифікатор має розмір від 50 КБ і багато чого іншого, а також уміє самонавчатися, заповнюючи пропуски у словнику (рис. 7.6).

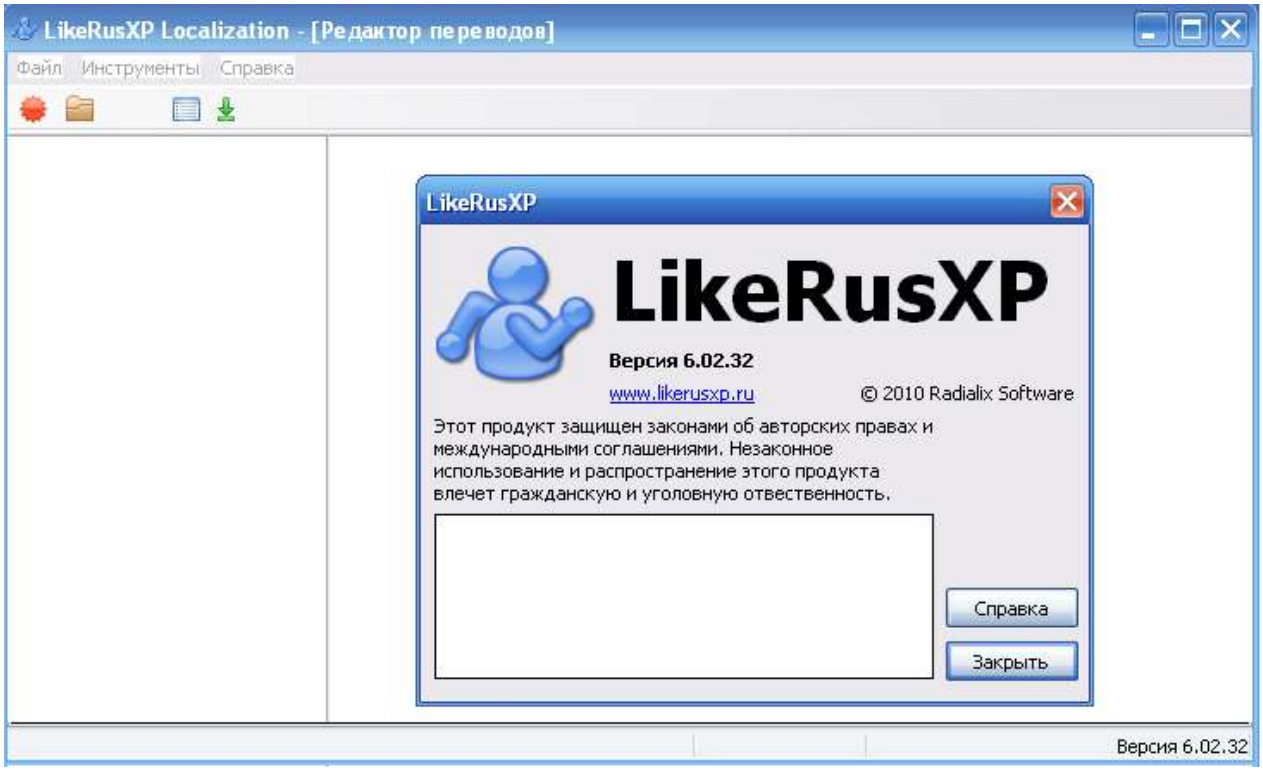


Рис. 7.6. **Головне вікно програми LikeRusXP**

LikeRusXP – це універсальний русифікатор (локалізатор) програмного забезпечення під Windows, що використовує для перекладу як словники, так і інтелектуальний рушій SmartLike.

Основні можливості:

автоматичний і ручний переклад інтерфейсу програми;

переклад рядків, зашитих у код програми;

створення кінцевого патча локалізації розміром від 35 КБ;

розширення словника;

використання необмеженої кількості словників зі своїми пріоритетами;
гнучка мова шаблонів;
переклад текстових файлів за шаблоном;
порівняння текстових файлів;
порівняння перекладеного і неперекладеного файлів.

Загальне завдання для виконання

1. Аналітична частина

Етап 1. Порівняльний аналіз структури DOS-заголовка файлів *.exe і *.dll

Проведіть дослідження DOS заголовка виконуваних файлів, імена яких задано в п. 1 індивідуального завдання. Урахуйте, що файли розташовуються в %SystemRoot%\System32. Результати дослідження занесіть до табл. 7.1. Після табл. 7.1 у звіті відобразіть уміст програми-заглушки DOS-заголовка.

Таблиця 7.1

Порівняння DOS-заголовка виконуваних файлів

№ п/п	RVA	Параметри	Значення	
			*.exe	*.dll
1	2	3	4	5
1	00000000	Сигнатура		
2	00000002	Довжина останньої неповної сторінки		
3	00000004	Довжина файла у сторінках		
4	00000006	Кількість елементів у таблиці переміщення		
5	00000008	Довжина заголовка		
6	0000000A	Мінімум необхідної пам'яті в кінці програми		
7	0000000C	Максимум необхідної пам'яті в кінці програми		
8	0000000E	Значення регістру SS		
9	00000010	Значення регістру SP		
10	00000012	Контрольна сума		
11	00000014	Значення регістру IP		
12	00000016	Значення регістру CS		
13	00000018	Зсув першого елемента таблиці переміщення		

Закінчення табл. 7.1

1	2	3	4	5
14	0000001A	Номер оверлея		
15	00000024	OEM-ідентифікатор		
16	00000026	OEM-інформація		
17	0000003C	Адреса Windows-заголовка		

* Замість символу (*) поставте ім'я заданого файла

Етап 2. Порівняльний аналіз структури Windows-заголовка файлів *.exe і *.dll

Проведіть дослідження Windows-заголовків виконуваних файлів, імена яких задано в п. 1 індивідуального завдання. Результати дослідження занесіть до табл. 7.2.

Таблиця 7.2

Порівняння Windows-заголовка виконуваних файлів

№ п/п	RVA	Параметри	Значення		Опис
			*.exe	*.dll	
1	2	3	4	5	6
1	...	Сигнатура			
Обов'язкова частина					
2	...	Процесор			
3	...	Кількість секцій			
4	...	Дата створення файла			
5	...	Зсув COFF-таблиці			
6	...	Кількість байтів COFF-таблиці			
7	...	Розмір IMAGE_OPTIONAL_HEADER			
8	...	Прапори			
Необов'язкова частина					
Стандартні поля					
9	...	Magic			
10	...	Версія лінкера			
11	...	Розмір коду			
12	...	Розмір ініціалізованих даних			
13	...	Розмір неініціалізованих даних			
14	...	Адреса точки входу у програму			
15	...	RVA-секції коду			

Закінчення табл. 7.2

1	2	3	4	5	6
16	...	RVA-секції даних			
Додаткові поля					
17	...	Базова адреса			
18	...	Межа вирівнювання секцій			
19	...	Розмір сектора на диску			
20	...	Версія ОС			
21	...	Версія виконуваного файла			
22	...	Версія підсистеми ОС			
23	...	Розмір завантаженого модуля			
24	...	Розмір заголовків			
25	...	Контрольна сума			
26	...	Тип підсистеми інтерфейсу			
27	...	Прапори завантаження Dll			
28	...	Розмір резервованого стека			
29	...	Розмір використовуваного стека			
30	...	Розмір резервованої купи			
31	...	Розмір використовуваної купи			
32	...	Кількість входів у масиві DataDirectory			

Етап 3. Порівняльний аналіз масиву секцій Windows-заголовка файлів *.exe і *.dll

Проведіть дослідження масиву секцій Windows-заголовка виконуваних файлів, імена яких задано в п. 1 індивідуального завдання. Результати дослідження занесіть до табл. 7.3.

Таблиця 7.3

Порівняння масиву секцій Windows-заголовка виконуваних файлів

№ п/п	Імена елементів масиву	*.exe		*.dll	
		VA	розмір	VA	розмір
1	2	3	4	5	6
1	Секція експорту				
2	Секція імпорту				
3	Секція ресурсів				
4	Секція винятків				
5	Секція безпеки				

Закінчення табл. 7.3

1	2	3	4	5	6
6	Секція налаштування адрес				
7	Секція налаштування				
8	Секція особливостей архітектури				
9	Секція таблиці значень, специфічних для мікропроцесора				
10	Секція локальної пам'яті потоку				
11	Секція завантаження конфігурації				
12	Bound Import Table				
13	Таблиця адрес імпорту				
14	Секція затримки завантаження імпорту				
15	Секція COM+				

Етап 4. Дослідження таблиць секцій файлів *.exe і *.dll

Проведіть дослідження таблиці секцій виконуваних файлів, імена яких задано в п. 1 індивідуального завдання. Результати дослідження занесіть до табл. 7.4.

Таблиця 7.4

Таблиця секцій виконуваних файлів

№ п/п	Ім'я	Virtual Size	RVA	Розмір секції	Зсув від початку файла	Прапори (характеристики)	Опис
Файл *.exe							
1	.text						
2	.data						
3	...						
Файл *.dll							
1	.text						
2	.data						
3	...						

Етап 5. Дослідження секції імпорту файлів *.exe і *.dll

Проведіть дослідження секції (.idata) виконуваних файлів, імена яких задано в п. 1 індивідуального завдання. Результати дослідження занесіть до табл. 7.5.

Таблиця 7.5

Секція імпорту виконуваних файлів

№ п/п	RVA-таблиці імен імпорту	Час створення	Forwarder Chain	Ім'я dll, що імпортується	RVA-таблиці адрес імпорту
1	2	3	4	5	6
Файл *.exe					
1				kernel32.dll	
2				user32.dll	
3				...	
Файл *.dll					
1				kernel32.dll	
2				user32.dll	
3				...	

Примітка. Імена бібліотек, що імпортуються, індивідуальні, і їхня кількість може перевищувати 3. Тоді в табл. 7.5 відобразіть дані про всі бібліотеки.

2. Прикладна частина

Проведіть русифікацію вказаних у п. 2 індивідуального завдання програм, послідовно використовуючи такі програми:

1. Resource Hacker.
2. Multilizer.
3. LikeRusXP.

Обов'язковими для модифікації є такі типи ресурсів: курсори, іконки, меню, діалогові вікна, таблиці рядків.

У звіті відобразіть:

скриншоти початкового стану ресурсних модулів, що підлягають модифікації;

скриншоти кінцевого стану ресурсних модулів, що підлягають модифікації;

порядок здійснення модифікації.

Окремо подайте чотири виконувані файли: початковий і три модифіковані відповідною програмою.

3. Програмна частина – додаткове завдання

Програмним способом проведіть дослідження версії виконуваного тестового файла, скомпонованої лінковником, відповідно у двох версіях: RELEASE і DEBUG. Виведіть у вікно **тільки** відмінні секції та конструкції, однакові секції не виводьте (табл. 7.6).

Таблиця 7.6

Індивідуальні завдання для виконання

Варіанти	Об'єкти дослідження	Об'єкти дослідження	Файли для русифікації
1	notepad.exe	vwipxspx.dll	Deskmon.exe
2	shrpubw.exe	w32topl.dll	Filemon.exe
3	charmap.exe	wldap32.dll	Portmon.exe
4	mmc.exe	xmlprov.dll	Regmon.exe
5	cliconfg.exe	wuapi.dll	Winhex.exe
6	colorcpl.exe	winhttp.dll	FileSci.exe
7	mobsync.exe	unimdmat.dll	LinkProcessPort.exe
8	narrator.exe	zipfldr.dll	FSalv.exe
9	eudcedit.exe	wiashext.dll	AVI Preview.exe
10	sigverif.exe	gptext.dll	Direct Show Filter Manager.exe
11	wscript.exe	winfax.dll	NTPower3.exe
12	iexpress.exe	uniplat.dll	Hcw.exe
13	magnify.exe	url.dll	Pt.exe
14	syskey.exe	usbui.dll	Netscan.exe
15	cleanmgr.exe	wzcsapi.dll	DiE.exe
16	msinfo32.exe	utildll.dll	amdcpuid.exe
17	msiexec.exe	wintrust.dll	exeinfope.exe
18	mstsc.exe	winntbbu.dll	APISpy32.exe
19	eventvwr.exe	version.dll	Tcpview.exe
20	credwiz.exe	vdmdbg.dll	Sysinfo.exe
21	cmstp.exe	vb5stkit.dll	ToolApp.exe
22	osk.exe	wow32.dll	TextApp.exe
23	verifier.exe	winipsec.dll	Gallery.exe
24	psr.exe	w32time.dll	Pendulum.exe
25	write.exe	mssap.dll	cpuz.exe
26	calc.exe	mssip32.dll	BrushApp.exe
27	dxdiag.exe	msrle32.dll	Regmon.exe
28	dialer.exe	msorc32r.dll	Heat.exe
29	mspaint.exe	digest.dll	PacketsDump.exe
30	MdSched.exe	mssign32.dll	LogAnalysisCenter.exe

Контрольні запитання

1. Що таке PE?
2. У чому полягає схожість і відмінність DOS- і Windows-заголовків?

3. Поясніть фізичний сенс параметрів DOS- і Windows-заголовків.

4. Розшифруйте значення поля характеристик для довільно вибраної секції виконуваного файла.

5. Яка специфіка таблиці об'єктів?

6. Поясніть, у чому полягають особливості секцій .idata, .edata, .data, .text та .rsrc?

7. Як називають частину виконуваного файла, що визначає початок заголовків?

8. Назвіть механізм відображення exe-файла на віртуальний адресний простір.

9. Розрахуйте значення року, коли настане обмеження на відображення всередині виконуваного файла формату дати та часу створення файла.

10. Укажіть граничне значення, яке підтверджує Windows-файл.

11. Що розміщено по RVA 40h в exe-файлі Windows?

12. На яку межу вирівнюються ці секцій, якщо exe-файл скомпоновано компонувальником фірми Microsoft?

13. Яке шістнадцяткове значення OS2 сигнатури?

14. Скільки байтів відводиться під зберігання імені секції?

15. Яке значення поля Machine показує, що виконуваний файл скомпільовано для захищеного режиму Intel?

16. Що становить собою вся таблиця секцій?

17. Яку дію виконує транслятор із символьними іменами?

18. Укажіть зсув, за яким можна знайти секцію виконуваного файла віртуального драйвера.

19. За яким RVA (у hex) у виконуваному файлі розташовується зсув у файлі першого елемента таблиці переміщення?

20. Назвіть атрибут, який властивий тільки секції .text виконуваного файла.

21. За фрагментом exe-файла 4D 5A 0A 00 02 00 00 00 04 00 0F FF FF 00 00 ... визначте розмір файла у КБ.

22. Який будуть мати вигляд останні 4 байти обов'язкової частини Windows-заголовка?

23. Назвіть розмір у байтах обов'язкової частини Windows-заголовка.

24. Скільки секцій може бути розміщено у виконуваному файлі?

Лабораторна робота 8
Дослідження бібліотек динамічного компонування

Мета роботи: ознайомлення з одним із найбільш важливих структурних елементів Windows-бібліотеками динамічного компонування, набуття практичних навичок у створенні динамічних бібліотек на основі явного або неявного їхнього завантаження.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно вивчити принципи динамічного з'єднання функцій і процедур Windows-додатків, звернути увагу на відмінність у використанні стандартних статичних бібліотек функцій мов програмування та динамічних бібліотек системного призначення.

Додаткову інформацію під час підготовки до роботи можна отримати в [2; 16; 19].

Теоретичні відомості

Бібліотеки динамічного компонування (Dynamic Link Libraries, DLL) або динамічні бібліотеки є одним із найбільш важливих структурних елементів Windows. Більшість файлів, із яких складається Windows, є або програмними модулями, або модулями бібліотек, які динамічно підключаються. Велика частина принципів, які належать для написання програм, повністю підходить і для написання цих бібліотек, проте є декілька важливих відмінностей.

Термін "динамічне з'єднання" (dynamic linking) належить до процесів, які Windows використовує для того, щоб пов'язати функції одних модулів із реальною функцією з модуля бібліотек.

Статичне з'єднання (static linking) має місце у процесі створення програми, якщо для створення виконуваного файла (.exe). пов'язуються воєдино всі об'єктні (.obj) файли та файли статичних бібліотек (.lib) і, переважно, скомпільовані файли опису ресурсів (.res). На відміну від цього, динамічне з'єднання має місце під час виконання програми.

Файли kernel32.dll, user32.dll, gdi32.dll, файли драйверів – усе це бібліотеки, які динамічно підключаються. Ці бібліотеки можна використовувати у всіх програмах Windows.

Бібліотеки, які динамічно підключаються, можуть містити тільки ресурси або дані й не містити програм. Хоча модуль бібліотеки, яка динамічно підключається, може мати будь-яке розширення (наприклад, .exe. fon), стандартним розширенням, прийнятим у Windows, є .dll. Тільки ті бібліотеки, які динамічно підключаються, мають розширення .dll. Windows може завантажити автоматично. Якщо файл має інше розширення, то програма має завантажити модуль бібліотеки явним чином. Для цього використано функцію LoadLibrary (LoadLibraryEx).

Переважно, найбільшого сенсу динамічні бібліотеки набувають у великому додатку, які використовують великий набір функцій, склад і зміст яких може змінюватися за час експлуатації та модифікації.

Бібліотеки статичного компонування (.lib) – це просто набір об'єктних COFF-файлів плюс деякі початкові секції, що дозволяють швидко встановлювати розташування потрібних об'єктних файлів, що містяться всередині бібліотеки. Документація за форматом LIB-файлів називає LIB-файли архівами.

Усі LIB-файли починаються з однієї й тієї самої 8-байтової сигнатури. Цю сигнатуру визначено у файлі winnt.h:

```
#define IMAGE_ARCHIVE_START "!<arch>\n"
```

Решта частини файла становить ряд записів змінної довжини. Кожен запис починається структурою IMAGE_ARCHIVE_MEMBER_HEADER:

```
typedef struct IMAGE ARCHIVE_MEMBER_HEADER {
BYTE  Name[16];
BYTE  Date[12];
BYTE  USERLD[6];
BYTE  GROUPLD[6];
BYTE  Mode[8];
BYTE  Size[10];
BYTE  EndHeader[2];
}IMAGE_ARCHIVE_MEMBER_HEADER
*PIMAGE_ARCHIVE_MEMBER_HEADER;
```

Кожна структура IMAGE_ARCHIVE_MEMBER_HEADER відповідає або об'єктному файлу всередині бібліотеки, або одному запису з невеликого набору спеціальних записів. Ці спеціальні записи перебувають на початку

бібліотеки та існують для того, щоб компонувальник надалі міг швидко відшукувати об'єктні файли в LIB-файлі.

Початкові дані для члена архіву слідують відразу за структурою IMAGE_ARCHIVE_MEMBER_HEADER, із якої починається кожний запис. Для більшості членів архіву записів початкові дані такі самі, як і в об'єктному файлі. Дійсно, коли програма здійснює виведення LIB-файлів, вона викликає ті самі процедури, що й під час оброблення об'єктного файла. Рис. 8.1 показує формат LIB-файлів.

Сигнатура !<arch>.
IMAGE_ARCHIVE_MEMBER_HEADER (/Y)
Дані першого члена компонувальника
IMAGE_ARCHIVE_MEMBER_HEADER (/Y)
Дані другого члена компонувальника
IMAGE_ARCHIVE_MEMBER_HEADER (/Y)
Дані члена Longnames
IMAGE_ARCHIVE_MEMBER_HEADER (FOO.OBJ/)
Дані об'єктного файла
IMAGE_ARCHIVE_MEMBER_HEADER (/104)
Дані об'єктного файла
. . .

Рис. 8.1. **COFF-формат LIB-файлів**

Слід розглянути поля структури IMAGE_ARCHIVE_MEMBER_HEADER.

BYTE Name[16]. Ім'я члена архіву. Якщо символ "/" з'являється після ASCII-рядка (наприклад, FOO.OBJ/), то рядок перед символом "/" становить ім'я члена. Якщо ім'я починається із символу "/", за яким слідує десяткове число (наприклад, /104), то число є зсувом імені члена архіву всередині

члена Longnames LIB-файла. У попередньому прикладі ім'я члена починається зі 104-го байта від початку області Longname. Є також спеціальні імена для спеціальних членів архіву:

```
#define IMAGE_ARCHIVE_LINKER MEMBER "/"
#define 1MAGE_ARCHIVE_LONGNAMES_MEMBER "//"
```

Для об'єктних файлів усередині бібліотеки імпорту це поле становить ім'я DLL, що містить функції, що імпортують:

BYTE Date[12]. Дата і час створення члена. Це число зберігається в десятковому ASCII-вигляді.

BYTE USERID[6]. Десяткове ASCII-подання ідентифікатора користувача. Мабуть, завжди є рядком NULL.

BYTE GROUPID[6]. Десяткове ASCII-подання ідентифікатора групи. Мабуть, завжди є рядком NULL.

BYTE Mode[8]. Десяткове ASCII-подання файлового режиму. Мабуть, завжди дорівнює нулю.

BYTE Size[10]. Розмір даних члена, поданий у десятковій ASCII-формі. Формат даних залежить від їхнього типу (указаний у вже описаному полі Name).

BYTE EndHeader[2]. ASCII рядок \n.

Члени компонувальника. Усякий LIB-файл має дві секції членів компонувальника, що виконують функцію змісту для решти частини файла. Обидва члени мають ім'я "/" і розрізняються за порядком проходження у файлі.

Перший член компонувальника – це перший член архіву з ім'ям "/", а другий член компонувальника – це другий член архіву з ім'ям "/".

Обидва члени компонувальника – це, насправді, переліки загальнодоступних символів у LIB-файлі разом зі зсувами всередині файла членів – об'єктних модулів, які містять загальнодоступні символи. Два члени компонувальника мають різні формати. Навіщо ж потрібно дві копії однакової інформації? Перший член компонувальника зберігає свою інформацію в тому порядку, у якому об'єктні модулі йдуть далі в LIB-файлі. Це призводить до неоптимальних пошуків. Другий член компонувальника зберігає свої символи в алфавітному порядку, що робить його набагато кориснішим для компонувальника. Відповідно до документації Microsoft, компонувальник ігнорує перший член компонувальника і завжди використовує другий член.

Перший член компонувальника має такий формат:

DWORD NumberOfSymbols. Кількість загальнодоступних символів у цій бібліотеці. Цю кількість подано у форматі big-endian (відображає COFF-формат для машин, відмінних від машин i386). Функція ConvertBigEndian здійснює перемикання з формату big-endian у формат little-endian, використовуваний i386.

DWORD Offsef[NumberOfSymbols]. Масив файлових зсувів інших членів архіву. Ці зсуви мають формат big-endian. Кожен із цих членів – це член типу OBJ. Кожен елемент цього масиву відповідає імені символу в переліку подальших рядків ASCII.

BYTE StringTable[?]. Це нерозривна серія рядків у стилі З в пам'яті.

Насправді, кожен елемент масиву Offset відповідає одному загальнодоступному символу, ім'я якого з'являється в області StringTable. Наприклад, третій елемент масиву Offsets відповідає третьому рядку в області StringTable:

```
First Linker Member:
Symbols:  00000006
MbrOffs   Name
--------  -----------
00000180  _DumpCAP@0
00000180  _StartCAP@0
00000180  _StopCAP@0 ...
```

Формат *другого члена компонувальника* заплутано із-за додавання масиву, необхідного для швидкого пошуку символів. Другий член компонувальника має такий формат:

DWORD NumberOfMembers. Це подвійне слово містить кількість членів в архіві об'єктних модулів, наступних у файлі.

DWORD Offsets[NumberOfSymbols]. Масив файлових зсувів інших членів архіву. На відміну від першого члена компонувальника, ці зсуви задано у природному форматі машини (тобто у форматі little-endian для i386).

DWORD NumberOfSymbols. Кількість загальнодоступних символів в масиві StringTable (отже, і кількість загальнодоступних символів у бібліотеці). Це поле до того ж містить кількість елементів у наступному масиві Indices.

WORD Indices[NumberOfSymbols]. Цей масив містить індекси (відлік починається з 1) масиву Offsets (описано на два поля раніше). Цей масив іде паралельно рядкам масиву StringTable.

BYTE StringTable[NumberOfSymbols]. Це нерозривна серія рядків у стилі мови C у пам'яті.

Для того щоб відшукати об'єктний файл за його символом, використовуючи другий член компонувальника, компонувальник спочатку переглядає масив StringTable та обчислює відносний індекс рядка в масиві. Потім компонувальник використовує цей індекс для пошуку слова в масиві Indices. Нарешті, компонувальник віднімає 1 із цього слова в масиві Indices і використовує результат як індекс масиву Offsets. Знайдене подвійне слово в масиві Offsets якраз і буде зсувом в об'єктному файлі, що містить загальнодоступний символ.

Член Longnames. Дані в секції архівного члена Longnames – це просто набір рядків у стилі мови C, що слідують одна за одною. Рядок поміщається в секцію Longnames, якщо вона дуже велика, щоб укластися в 16 байтів, зарезервованих для поля Name у структурі IMAGE_ARCHIVE_MEMBER_HEADER. У цьому разі поле Name містить символ "/", за яким слідує десяткове ASCII-подання зсуву рядка в секції Longnames.

Програма DLL2Lib.exe. Це сервісна програма перетворить DLL-файл на його еквівалент статичного файла бібліотеки. Після цього можна перемістити справжній файл DLL зі статичним файлом бібліотеки, перебудувати цей додаток і поширювати із DLL-файлом. Найважливіша особливість це те, що процес оброблення не потребує яких-небудь джерел коду DLL-файлів. DLL to Lib інтегрується з багатьма утилітами, включаючи Import Library Reference Information Generator, Symbol Finder тощо, для того щоб користувач був упевнений, що процес конверсії успішний (рис. 8.2).

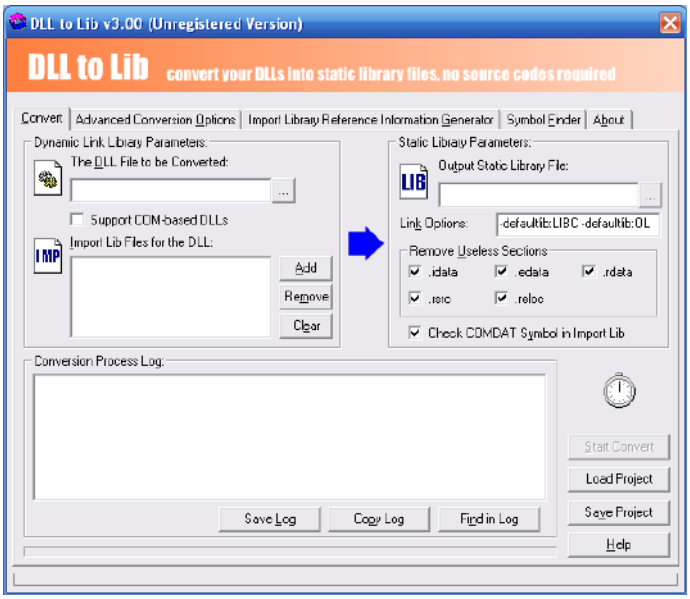


Рис. 8.2. **Головне вікно програми DLL2Lib**

Програма ProcessInfo.exe. Ця програма демонструє, як створити дуже корисну утиліту на основі ToolHelp-функций. Після запуску ProcessInfo відкривається діалогове вікно (рис. 8.3).

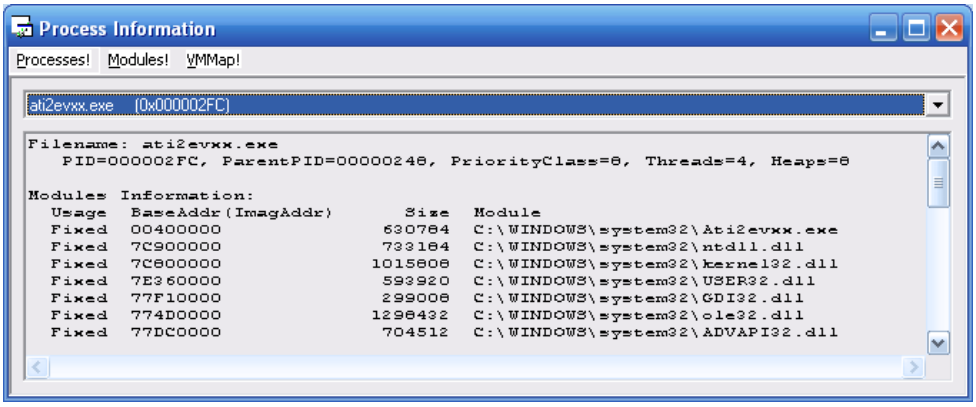


Рис. 8.3. **Вікно Processes! програми ProcessInfo**

ProcessInfo спочатку перелічує всі процеси, що виконують у системі, а потім виводить у верхній список, що розкривається, імена та ідентифікатори кожного процесу. Далі вибирає перший процес та інформацію про нього показує. у великому текстовому полі, доступному тільки для читання. Для поточного процесу повідомляє його ідентифікатор (разом з ідентифікатором батьківського процесу), клас пріоритету і кількість потоків, що виконують зараз у контексті процесу.

В інформацію про модулі входить список усіх модулів (EXE- і DLL-файлів), спроектованих на адресний простір поточного процесу. За фіксований модуль (fixed module) уважають той, який був неявно завантажений під час ініціалізації процесу. Для явно завантажених DLL показує лічильники кількість користувачів цих DLL. У другому стовпці виведено базову адресу пам'яті, на яку спроектовано модуль. Якщо модуль розміщено не за заданою для нього базовою адресою, у дужках з'являється й ця адреса. У третьому стовпці повідомлено розмір модуля в байтах, а в останньому – повне (разом зі шляхом) ім'я файла цього модуля. І, нарешті, унизу показано інформацію про потоки, що виконують у цей момент у контексті поточного процесу. Водночас відображає ідентифікатор потоку (Thread ID, TID) і його пріоритет.

На додаток до інформації про процеси можна вибрати елемент меню Modules. Це змусить ProcessInfo перелічити всі модулі, завантажені в системі, і помістити їхні імена у верхній список. Далі ProcessInfo вибирає перший модуль і виводить інформацію про нього (рис. 8.4).

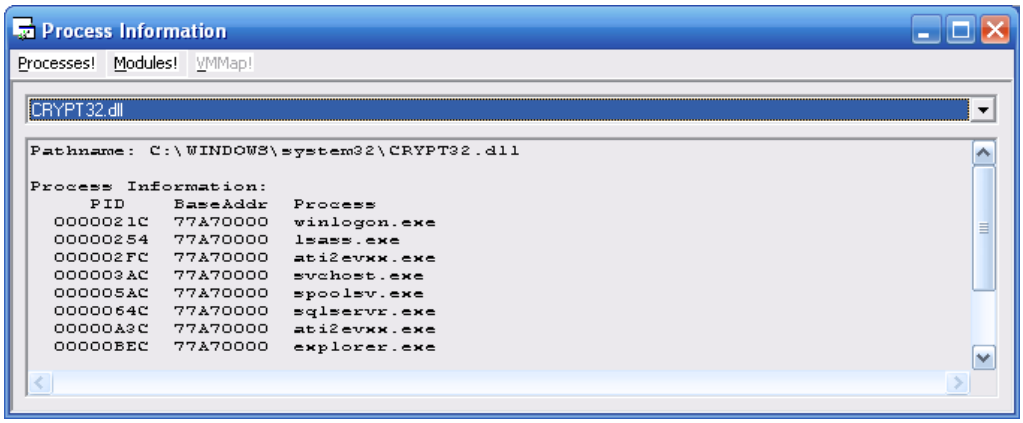


Рис. 8.4. **Вікно Modules! програми ProcessInfo**

У цьому режимі утиліта ProcessInfo дозволяє легко визначити, у яких процесах задіяно цей модуль. Повне ім'я модуля з'являється у верхній частині текстового поля, а в розділі Process Information перелічено всі процеси, що містять цей модуль. Там же показано ідентифікатори й імена процесів, у які завантажено модуль і його базові адреси в цих процесах.

Аналіз атак перехоплення DLL. Перехоплення DLL є можливим, оскільки практично всі додатки Windows у своїй роботі покладаються на бібліотеки DLL як частину їхньої основної функціональності. DLL містять модульні компоненти коду, які розробники можуть використовувати всередині додатка для виконання різних функцій. Сама Windows покладається на такий самий тип архітектури та містить безліч DLL, що виконують різноманітні функції.

Окрім DLL-файлів, інтегрованих у Windows, розробники програм часто створюють власні DLL, що містять функції, які використовуються програмою. Коли це відбувається, створені розробником DLL-файли пакують і встановлюють разом із додатком. Проблема полягає в тому, як додаток завантажує DLL. За замовчуванням, якщо для додатка не вказано статичний шлях до DLL, потрібний для цього додатка, він здійснює процес його динамічного пошуку. Виконуючи цю процедуру, додаток спочатку виконує пошук у каталозі, із якого він виконувався, а потім виконує пошук у системному каталозі, 16-розрядному системному каталозі, каталозі Windows, у поточному каталозі, а потім у каталогах, перелічених у змінному середовища PATH операційної системи. Під час пошуку цими шляхами додаток буде використовувати перший DLL, який він знайде.

Знаючи це, уявіть собі ситуацію, у якій створено додаток, що потребує динамічного пошуку необхідного DLL-файла під час завантаження. Додаток негайно виконує пошук шляхом, із якого його було виконано і знаходить відповідний DLL. Через нещастя для кінцевого користувача, дійсний файл DLL, пов'язаний із додатком, розташовано в системному каталозі Windows. DLL-файл, поміщений у каталог із додатком, є файлом, який було змінено зловмисником і який дозволяє віддалено виконувати командну оболонку в системі. Звичайно, додаток ніколи не отримає дійсний файл DLL, оскільки він уже знайшов необхідний файл.

Визначення вразливих додатків. Основною проблемою з атаками перехоплення DLL є те, що компанія Microsoft не може випустити єдине виправлення для всіх уразливих додатків. Це неможливо тому, що це просто може призвести до некоректної роботи (або повної відмови в роботі) деяких додатків. Є два способи визначення вразливості використовуваних додатків.

Першим способом є перевірка публічних ресурсів, викладених дослідниками у сфері безпеки, які вже виконали основну роботу. Одним із таких корисних ресурсів є ресурс http://www.exploit-db.com/dll-hijacking-vulnerable-applications, що містить досить великий список (рис. 8.5).

Рис. 8.5. **Список уразливих додатків Exploit DB**

Другий спосіб потребує більше роботи, але має виконуватися в середовищах із високим ступенем безпеки для пошуку вразливих додатків у системі.

Цей набір називають **DllHijackAuditKitv2** і його можна знайти http://blog.metasploit.com/2010/08/better-faster-stronger.html. Після завантаження необхідно зайти в систему із правами адміністратора, витягнути компоненти із ZIP-файла і виконати **01_StartAudit.bat.** Цей сценарій завантажить монітор Sysinternals Process Monitor і почне перевірку системи на вразливість додатків.

Примітка. Завантаження монітора процесів частіше виявляється неуспішним, ніж успішним, тому якщо таке відбувається під час виконання сценарію, можна вручну завантажити монітор Process Monitor. Після завантаження важливо переконатися, що Process Monitor розташовано в тому самому каталозі, що й сценарії аудиту. Для запуску початкового сценарію аудиту буде потрібно певний час від 15 хв до 1 год (рис. 8.6).

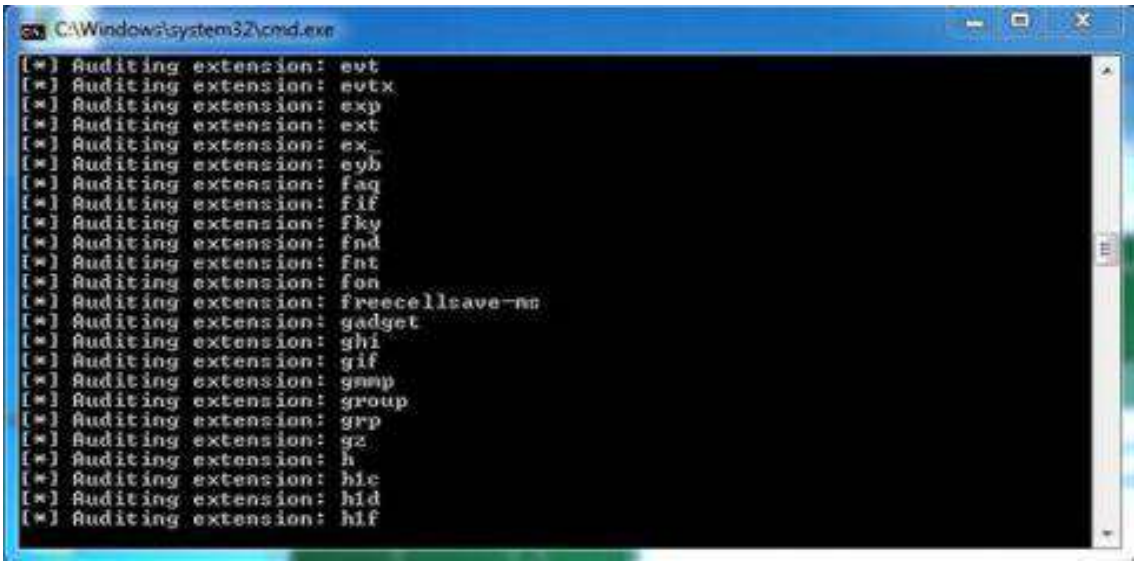


Рис. 8.6. **Аудит додатків, пов'язаних із певними файлами**

Після завершення роботи сценарію аудиту необхідно перейти на додаток монітора процесів, що був ініційований сценарієм, і зберегти звіт, що він згенерував. Для цього натиснути *Файл* і обрати пункт *Зберегти*. Обов'язково збережіть файл у **форматі CSV** з ім'ям Logfile.CSV у каталозі набору аудиту.

Далі запустіть **сценарій 02_Analyze.bat.** Цей сценарій проаналізує CSV-файл і визначить потенційні вразливості. Якщо вразливість буде

знайдено, додаток автоматично згенерує код засобу атаки, який можна використовувати для демонстрації вразливості (рис. 8.7).

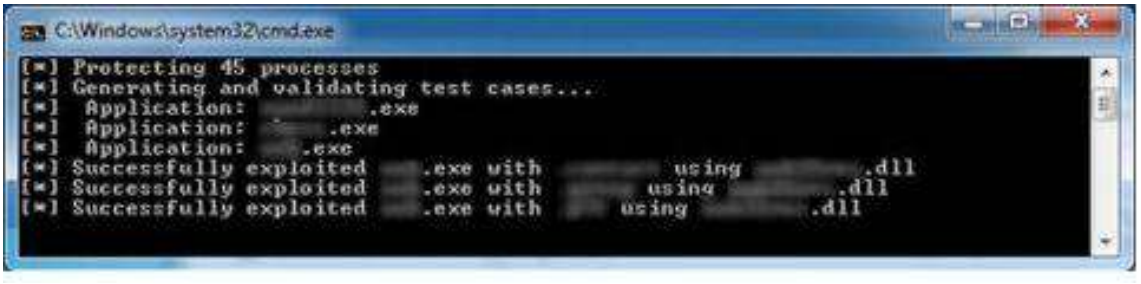


Рис. 8.7. **Другий сценарій, що автоматично намагається використовувати виявлені вразливості**

Після закінчення інтерпретатор команд, залишений відкритим сценарієм, має видати список додатків, які були успішно використані для атаки. Для кожного такого додатка сценарій створить підкаталог у каталозі засобів атаки. Ці підкаталоги містять робочі засоби атаки для відповідних додатків. У недобрих руках ці засоби атаки можуть робити такі жахливі речі, як запуск командних оболонок і прослуховування чорного ходу (back door listeners). Це просто приклади можливості використання засобів атаки, тому вони запускають додаток калькулятора (calc.exe), щоб продемонструвати наявність уразливості.

Наявність такої кількості вразливостей для перехоплення DLL становить цікавий випадок, оскільки його нелегко виправити за допомогою корегувань операційної системи, і воно впливає на велику кількість широко використовуваних додатків. Кращим захистом у цій ситуації є знання того, як працює вразливість, як перевіряти наявність уразливих додатків і як надавати правильну інформацію тим людям, які можуть виправити проблему.

Загальне завдання для виконання

1. Аналітична частина

Етап 1. Проведіть дослідження **7** динамічних бібліотек, розташованих у %SYSTEMROOT%\System32, указаних у табл. 8.6 індивідуального

завдання. У результаті дослідження подайте у звіті заповнену табл. 8.1 такого змісту (бібліотеку A2Nusd.dll вибрано для прикладу):

Таблиця 8.1

Характеристики динамічних бібліотек

№ п/п	Бібліотеки	Кількість функцій, що експортуються	Кількість бібліотек, що імпортуються	Назви бібліотек, що імпортуються	Розмір бібліотек, у байтах
1	A2Nusd.dll	4	7	kernel32.dll	989 696
				user32.dll	577 536
				gdi32.dll	278 016
				advapi.dll	678 104
				comctl32.dll	921 088
				ole32.dll	1 281 024
				setupapi.dll	990 208
				Усього:	5 715 672
…	…	…	…	…	…
7	...	...	...	...	...

Етап 2. Проведіть дослідження використання процесами **семи** динамічних бібліотек, розташованих у %SYSTEMROOT%\System32, указаних у в табл. 8.6 індивідуального завдання, за допомогою тестової програми ProcessInfo. У результаті дослідження подайте у звіті заповнену табл. 8.2 такого змісту (бібліотеку A2Nusd.dll вибрано для прикладу):

Таблиця 8.2

Використання динамічних бібліотек в операційній системі

№ п/п	Назви модулів	Інформація про бібліотеку (Process Information)		
		PID	BaseAddr	Process
1	A2Nusd.dll	00000A5C	7C630000	explorer.exe
		00000850	7C630000	TOTALCMD.EXE
...	…	…	…	…
7	...	...	...	...

2. Дослідна частина

Виконайте розроблення в середовищі Visual Studio двох проектів створення динамічної бібліотеки, яка б містила функцію обчислення, відповідно до п. 1 індивідуального завдання:

1. **DllCpp.dll** – із використанням засобів мови програмування C++ на Win32 API.

2. **DllCCs.dll** – із використанням засобів мови програмування C#.

У ході розроблення проектів використовуйте **однакові** назви змінних, які передаються в бібліотеку і повертаються з неї. Окрім указаних бібліотек, для дослідження необхідно вибрати створені під час роботи компонувальника відповідні файли статичної бібліотеки (DllCpp.lib, DllCs.lib).

Етап 3. Дослідження бібліотечного (lib) файла

Проведіть порівняльне дослідження структури lib-файлів за допомогою програми **PEView.exe**, які були створені в результаті компонування двох проектів: **DllCpp.lib** і **DllCCs.lib**. За потреби скористатися програмою **DLL2Lib.exe** для створення lib-файла. Результати порівняння занесіть до табл. 8.3.

Таблиця 8.3

Дослідження lib-файлів

№ п/п	RVA	Параметри	Значення		Опис
			DllCpp.lib	DllCCs.lib	
1	2	3	4	5	6
1	...	IMAGE_ARCHIVE_START			
IMAGE_ARCHIVE_MEMBER_HEADER					
2	...	Name			
3	...	Date			
4	...	USERLD			
5	...	GROUPLD			
6	...	Mode			
7	...	Size			
8	...	EndHeader			

Закінчення табл. 8.3

1	2	3	4	5	6
IMAGE_ARCHIVE_LINKER_MEMBER					
9	...	Offset Table			
10	...	Symbol Table			
IMAGE_ARCHIVE_MEMBER_HEADER					
11	...	Name			
12	...	Date			
13	...	USERLD			
14	...	GROUPLD			
15	...	Mode			
16	...	Size			
17	...	EndHeader			
IMAGE_ARCHIVE_LINKER_MEMBER					
18	...	Offset Table			
19	...	Symbol Table			
IMAGE_ARCHIVE_MEMBER_HEADER					
20	...	Name			
21	...	Date			
22	...	USERLD			
23	...	GROUPLD			
24	...	Mode			
25	...	Size			
26	...	EndHeader			
IMAGE_ARCHIVE_LINKER_MEMBER					
27	...	Offset Table			
28	...	Symbol Table			

За наслідками дослідження зробіть висновки, основну увагу у яких приділіть ступеню схожості (відмінності) розміщення функцій, що експортуються, змінних тощо.

Етап 4. Дослідження динамічної бібліотеки (dll) файла

Проведіть порівняльне дослідження структури dll-файлів за допомогою програми **PEView.exe**, які були створені в результаті компонування двох проектів: **DllCpp.dll** і **DllCCs.dll**. Результати порівняння занесіть до табл. 8.4 і 8.5.

Таблиця 8.4

Порівняння Windows-заголовка динамічних бібліотек

№ п/п	RVA	Параметри	Значення		Опис
			DllCpp.dll	DllCCs.dll	
1	...	Сигнатура			
Обов'язкова частина					
2	...	Процесор			
3	...	Кількість секцій			
4	...	Дата створення файла			
5	...	Зсув COFF-таблиці			
6	...	Кількість байтів COFF-таблиці			
7	...	Розмір IMAGE_OPTIONAL_HEADER			
8	...	Прапорці			

Таблиця 8.5

Таблиця секцій динамічних бібліотек

№ п/п	Ім'я	Virtual Size	RVA	Розмір секції	Зсув від початка файла	Прапорці (характеристики)	Опис
Файл DllCpp.dll							
1	.text						
2	.data						
3	...						
Файл DllCCs.dll							
1	.text						
2	.data						
3	...						

За наслідками дослідження зробіть висновки, основну увагу у яких приділіть ступеню схожості (відмінності) розміщення функцій, що експортуються, змінних тощо.

3. Прикладна частина

Етап 5. Здійснення аналізу атак перехоплення DLL-файлів

Визначте вразливі додатки в операційній системі за допомогою другого способу, використовуючи набір **DllHijackAuditKitv2** із програмою

Process Monitor. Це завдання необхідно виконувати із правами адміністратора. У результаті виконайте два файли сценарію:

01_ StartAudit.bat (подайте викладачеві у звіті на перевірку файл Logfile.CSV, завантаживши його в Microsoft Excel).

02_ Analyze.bat (подайте викладачеві у звіті код засобу атаки на вразливості операційної системи).

4. Програмна частина – додаткове завдання

Створіть виконуваний файл – додаток, у якому використано створену в п. 1 індивідуального завдання динамічну бібліотеку, яка реалізовує індивідуальний варіант (п. 2) приєднання DLL до програми. Усередині DLL має містити набори функцій, що реалізовують це індивідуальне завдання (табл. 8.6).

Таблиця 8.6

Індивідуальні завдання для виконання

Варіанти	Файли для аналітичної частини				Файли для дослідної частини	
1	2	3	4	5	6	7
1	activeds.dll	davclnt.dll	ipsecsvc.dll	mtxclu.dll	psbase.dll	seclogon.dll
2	actxprxy.dll	dhcpcsvc.dll	kerberos.dll	mydocs.dll	pstorsvc.dll	secur32.dll
3	apphelp.dll	dnsapi.dll	lmhsvc.dll	nddeapi.dll	rasadhlp.dll	sensapi.dll
4	basesrv.dll	drprov.dll	mlang.dll	netman.dll	rasman.dll	shdocvw.dll
5	adsldpc.dll	dimsntfy.dll	ksuser.dll	ncobjapi.dll	pxc40pma.dll	security.dll
6	browselc.dll	dssenh.dll	mpr.dll	netshell.dll	rasppp.dll	shfolder.dll
7	comsvcs.dll	hid.dll	msimg32.dll	odbc32.dll	rtutils.dll	tapisrv.dll
8	colbact.dll	esent.dll	msftedit.dll	ntlsapi.dll	riched32.dll	ssdpsrv.dll
9	crypt32.dll	icaapi.dll	msprivs.dll	ole32.dll	samsrv.dll	termsrv.dll
10	csrsrv.dll	iphlpapi.dll	msxml3.dll	psapi.dll	scrrun.dll	upnp.dll
11	browseui.dll	eapolqec.dll	msacm32.dll	netui1.dll	rastapi.dll	shlwapi.dll
12	browser.dll	duser.dll	mprapi.dll	netui0.dll	rasqec.dll	shimeng.dll
13	comdlg32.dll	gdi32.dll	msi.dll	ntshrui.dll	rpcss.dll	sxs.dll
14	cryptui.dll	imagehlp.dll	msv1_0.dll	pjlmon.dll	schannel.dll	umpnpmgr.dll

Закінчення табл. 8.6

1	2	3	4	5	6	7
15	atl.dll	dnsrslvr.dll	localspl.dll	netapi32.dll	rasapi32.dll	setupapi.dll
16	credui.dll	hnetcfg.dll	msonpmon.dll	odbcint.dll	samlib.dll	tcpmon.dll
17	authz.dll	dot3dlg.dll	midimap.dll	netlogon.dll	rasdlg.dll	sfc_os.dll
18	comctl32.dll	eventlog.dll	msgina.dll	ntmarta.dll	rpcrt4.dll	stobject.dll
19	certcli.dll	eappcfg.dll	msasn1.dll	normaliz.dll	rastls.dll	shsvcs.dll
20	batmeter.dll	dsound.dll	modemui.dll	netrap.dll	rasmans.dll	shell32.dll
21	clusapi.dll	ersvc.dll	mscoree.dll	ntdsapi.dll	resutils.dll	srvsvc.dll
22	cnbjmon.dll	es.dll	msctf.dll	ntlanman.dll	riched20.dll	ssdpapi.dll
23	comres.dll	hal.dll	msidle.dll	oakley.dll	rsaenh.dll	tapi32.dll
24	cryptdll.dll	ieframe.dll	mstlsapi.dll	oleaut32.dll	scecli.dll	themeui.dll
25	cryptsvc.dll	iertutil.dll	msutb.dll	onex.dll	scesrv.dll	trkwks.dll
26	cscui.dll	inetpp.dll	mswsock.dll	profmap.dll	sclgntfy.dll	uniplat.dll
27	advapi32.dll	dmserver.dll	linkinfo.dll	ncprov.dll	qutil.dll	sens.dll
28	clbcatq.dll	eappprxy.dll	mscms.dll	ntdll.dll	regapi.dll	spoolss.dll
29	audiosrv.dll	dot3api.dll	lsasrv.dll	netcfgx.dll	raschap.dll	sfc.dll
30	cscdll.dll	imm32.dll	msvcrt.dll	powrprof.dll	schedsvc.dll	unimdmat.dll

Варіант № 1

1. Створіть динамічну бібліотеку, що містить функції перетворення рядка символів із великих у малі.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 2

1. Створіть динамічну бібліотеку математичних функцій: $y_1(X) = X_1 + + X_2 + ... + X_n$ та $y_2(X) = X_1^1 + X_2^2 + ... + X_n^n$.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 3

1. Створіть динамічну бібліотеку функцій пошуку максимуму масиву цілих чисел. У функцію передано покажчик на масив і кількість елементів масиву.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 4

1. Створіть динамічну бібліотеку, що містить дві функції, які обчислюють довільний корінь зазначеного виразу.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 5

1. Створіть динамічну бібліотеку, що містить функцію, що обчислює числа Фібоначчі.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 6

1. Створіть динамічну бібліотеку, що містить функцію, яка обчислює n-факторіал.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 7

1. Створіть динамічну бібліотеку функцій обчислення суми довільного статичного ряду, причому цій функції передано значення як кількості членів ряду, так і їхній ступінь.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 8

1. Створіть динамічну бібліотеку, що містить функції перетворення рядка символів з англійських букв у відповідні за розкладкою клавіатури російські (наприклад, W → Ц; G → П тощо).

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 9

1. Створіть динамічну бібліотеку математичних функцій обчислення квадратних або кубічних коренів виразу, ступінь якого передано як параметр.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 10

1. Створіть динамічну бібліотеку, що містить дві функції, які обчислюють cos(x) і sin(x).

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 11

1. Створіть динамічну бібліотеку, що здійснює в одновимірному масиві пошук суми елементів масиву, розміщених до мінімального елемента.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 12

1. Створіть динамічну бібліотеку, що здійснює пошук у заданому текстовому фрагменті кількості повторів заданої букви. Як параметр передано фрагмент тексту й буква, кількість повторів якої у тексті потрібно знайти.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 13

1. Створіть динамічну бібліотеку, що здійснює у двовимірному масиві A операцію множення одного рядка на другий. Таким чином, як відповідь буде одновимірний масив C (наприклад, $c_1 = a_{11} \times a_{21}$, $c_2 = a_{12} \times a_{22}$, $c_3 = a_{13} \times a_{23}$ тощо).

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 14

1. Створіть динамічну бібліотеку, що здійснює в одновимірному масиві пошук суми елементів масиву, розміщених між мінімальним і максимальним елементами.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 15

1. Створіть динамічну бібліотеку, що здійснює в одновимірному масиві пошук суми елементів масиву з парними номерами.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 16

1. Створіть динамічну бібліотеку функцій пошуку мінімуму масиву цілих чисел. Параметрами є покажчик на масив і кількість елементів масиву.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 17

1. Створіть динамічну бібліотеку, що містить дві функції, які обчислюють tg(x) і ctg(x).

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 18

1. Створіть динамічну бібліотеку, що здійснює в цій прямокутній матриці цілих чисел суму елементів у тих рядках, що містить хоча б один негативний елемент.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 19

1. Створіть динамічну бібліотеку, що здійснює для заданої матриці розміром 6×6 такі *k*, що *k*-й рядок матриці збігається з *k*-м стовпцем і суму елементів у тих рядках, які містять хоча б один негативний елемент.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 20

1. Створіть динамічну бібліотеку, що містить функції перетворення рядка символів із малих у великі.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 21

1. Створіть динамічну бібліотеку, що здійснює в *n*-вимірному масиві пошук *n* максимальних елементів у кожному рядку.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 22

1. Створіть динамічну бібліотеку, що здійснює пошук середнього арифметичного значення елементів одновимірного масиву.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 23

1. Створіть динамічну бібліотеку, що здійснює пошук значення середньоквадратичного відхилення елементів одновимірного масиву.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 24

1. Створіть динамічну бібліотеку математичних функцій:

$y_1(X) = 1 / X_1 + 1 / X_2 + ... + 1 / X_n$ та $y_2(X) = 1 / X_1^1 + 1 / X_2^2 + ... + 1 / X_n^n$.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 25

1. Створіть динамічну бібліотеку, що здійснює для заданої матриці, переставляючи стовпці, відсортувати за зростанням елементів її останнього рядка.

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 26

1. Створіть динамічну бібліотеку, що здійснює сортування довільного масиву. Параметрами є покажчик на масив і кількість елементів масиву.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 27

1. Створіть динамічну бібліотеку, що містить дві функції, які обчислюють tg(x) і ctg(x).

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 28

1. Створіть динамічну бібліотеку, що здійснює в одновимірному масиві пошук суми елементів масиву з непарними номерами.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Варіант № 29

1. Створіть динамічну бібліотеку, що містить функції перетворення рядка символів із російських букв у відповідні за розкладкою клавіатури англійські (наприклад, Ц → W; П → G і т.д.).

2. Метод підключення DLL із використанням явного завантаження DLL.

Варіант № 30

1. Створіть динамічну бібліотеку, що здійснює в одновимірному масиві пошук суми елементів масиву, розміщених між першими й останнім нульовими елементами.

2. Метод підключення DLL із використанням неявного лінкування із DLL.

Контрольні запитання

1. Для чого використовують бібліотеки?
2. Поясніть різницю між динамічним і статичним з'єднаннями.
3. У чому полягає специфіка бібліотек, які динамічно підключаються?
4. Із якою метою використовують DLL, що розподіляють пам'ять?
5. Чим відрізняється динамічне з'єднання без імпорту?
6. Наведіть приклади системних DLL.
7. Опишіть процес створення DLL.
8. Поясніть структуру бібліотеки статичного компонування.
9. Яку роль відведено для сигнатури LIB-файла?
10. Назвіть особливості потреби використання в LIB-файлі двох членів компонувальника.
11. Який член компонувальника зберігає свої символи в алфавітному порядку?
12. Назвіть основне призначення архівного члена Longnames в LIB-файлі.
13. Що означає параметр BaseAddr, який визначають у процесі аналізу модулів динамічної бібліотеки?
14. Яким чином використовують динамічні бібліотеки під час здійснення атаки на операційну систему?
15. У чому полягає аудит додатків на їхні вразливості?
16. Поясніть, як у структурі виконавчого файла відображено кількість бібліотек, що ним імпортуються?

17. Поясніть, як у структурі виконавчого файла відображено кількість функцій, що ним експортуються?

18. У яких одиницях вимірюють параметр Offset Table LIB-файла?

19. Яким теоретично можливим числом обмежено кількість секцій динамічної бібліотеки?

20. Назвіть мінімальні компоненти, необхідні для компанувальника під час його роботи щодо створення динамічної бібліотеки.

21. Поясніть способи створення статичної бібліотеки.

22. За допомогою якого структурного елемента dll-файла стає можливим імпортування функцій із неї?

23. Скільки потрібно завдати проходів компанувальнику, щоб створити динамічну бібліотеку?

24. Як можна проаналізувати наявність у системі вразливостей через використання динамічних бібліотек?

Лабораторна робота 9
Дослідження системного реєстру ОС Windows

Мета роботи: вивчення структури ключів реєстру, типів параметрів ключів, способів редагування реєстру; набуття практичних навичок у роботі із системним реєстром ОС Windows.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно вивчити порядок створення, видалення, запису і читання даних із використанням, відповідно до власного варіанта, тих або тих ключів реєстру, параметрів ключів, способів редагування реєстру, а також під час підготовки до лабораторної роботи необхідно ознайомитися з теоретичним описом системного реєстру ОС Windows.

Додаткову інформацію під час підготовки до роботи можна отримати в [4; 19].

Теоретичні відомості

Реєстр – це база даних, яка є частиною операційної системи Windows. Реєстр зберігає інформацію, використовувану для ініціалізації й конфігурації додатків та управляє доступом до неї. Усі технології на платформі

Windows інтенсивно використовують реєстр. Слід розглянути основні групи відомостей, що зберігають у реєстрі:

програми установлення. Будь-яка грамотно написана програма під Windows повинна мати свій інсталятор-установник. Це може бути вбудований до ОС Microsoft Installer або будь-який інший. У будь-якому разі інсталятор використовує реєстр для зберігання своїх налаштувань, дозволяючи правильно встановлювати та видаляти додатки, не чіпаючи спільно використовувані файли;

розпізнавач. Під час кожного запуску комп'ютера програма ntdetect.com і ядро Windows розпізнає обладнання та зберігає цю інформацію в реєстрі;

ядро ОС. Зберігає багато відомостей у реєстрі про свою конфігурацію, зокрема й дані про порядок завантаження драйверів пристроїв;

диспетчер PnP (Plug and Play). Абсолютно необхідна річ для більшості користувачів, яка позбавляє їх від мук зі встановлення нового обладнання. Він зберігає свою інформацію в реєстрі;

драйвери пристроїв. Зберігають тут свої параметри;

адміністративні засоби. Наприклад, такі, як Панель управління, MMC (Microsoft Management Console) та ін.;

призначені для користувача профілі. Це ціла група параметрів, унікальна для кожного користувача: налаштування графічної оболонки, мережевих з'єднань, програм і багато чого іншого;

апаратні профілі. Дозволяють створювати декілька конфігурацій із різним обладнанням;

загальні налаштування програм. У кожного користувача є профіль, де зберігаються його налаштування для відповідної програми.

Структура реєстру

Формат відображення даних у системному реєстрі схожий на те, як папки й документи відображаються у провіднику Windows Explorer. Частково це пов'язано з тим, що структура системного реєстру подібна до структури каталогів. Якщо вдатися до термінології, використовуваної в системному реєстрі, то еквівалентом папки або каталогу тут є ключ (key), а документу або файлу відповідає параметр із його значенням (value). Фактично це означає, що реєстр має ієрархічну структуру та складається з розділів і пар *параметр – значення*.

На верхньому рівні цієї ієрархії перебувають кореневі розділи (root keys). Їх перелічено в табл. 9.1.

Таблиця 9.1

Кореневі розділи реєстру

№ п/п	Розділи (абревіатури)	Опис
1	HKEY_CLASSES_ROOT (HKCR)	Цей розділ містить визначення типів документів, зв'язків із файлами та інтерфейсу командного процесора
2	HKEY_CURRENT_USER (HKCU)	За допомогою цього ключа дістають доступ до призначених для користувача конфігурацій, програмного забезпечення
3	HKEY_LOCAL_MACHINE (HKLM)	Зберігає апаратні конфігурації, мережеві протоколи та класи програмного забезпечення
4	HKEY_USERS (HKU)	Використовують для зберігання вибраних користувачами глобальних параметрів, а також параметрів налаштування робочого столу
5	HKEY_CURRENT_CONFIG (HKCC)	Цей ключ установлює зв'язок із ключем відображення, що входить до складу підключа вибраної конфігурації config, HKLM, що міститься у ключі

Кожен із перелічених розділів містить інші розділи – аналоги файлової системи. Registry має структуру дерева. Кінцевим елементом дерева реєстру є **ключі** або **параметри**. Ключі, які містяться в інших ключах, називають **підключами** цього ключа. Усі ключі й параметри в межах підключа повинні мати унікальні імена.

HKEY_CLASSES_ROOT. Структура розділу дещо відрізняється від усіх останніх. Для кожного зареєстрованого розширення файла є підключ (наприклад .bmp).

Значення цього ключа за замовчуванням указує на підключ опису документа (ACDC_BMP), який розташований у тій же гілці основного розділу. У підключі опису документа й міститься ланцюжок ключів, що зберігають інформацію про асоціації, OLE, DDE.

HKEY_LOCAL_MACHINE. Інформація, збережена тут, використовується додатками, пристроями й системою і не залежить від того, хто був заявлений як користувач.

Пристрої можуть поміщати інформацію в системний реєстр за допомогою Plug&Play-інтерфейсу, програмні засоби – за допомогою стандартного API. HKEY_LOCAL_MACHINE містить ряд підрозділів, описаних у табл. 9.2.

Таблиця 9.2

Склад основного розділу HKEY_LOCAL_MACHINE

Розділи	Призначення
Hardware	Інформація про послідовні інтерфейси й модеми, які використовуються програмою HyperTerminal
SAM	Security Accounts Manager – адміністратор облікових даних у системі захисту (доступ до розділу блоковано операційною системою)
Security	Інформація про те, який комп'ютер у мережі стежить за безпекою мережі та чи підтримує (допускає) цей комп'ютер віддалене управління
Software	Інформація про програмні засоби, установлені на цьому комп'ютері, та різні конфігураційні дані програм
System	Інформація цього розділу управляє запуском системи, завантаженням драйверів пристроїв, сервісом Windows і поведінкою системи

Параметри є в кожного ключа та підключа. Параметри складаються з трьох частин: ім'я параметра, тип параметра та його значення. Допустимі типи параметрів показано в табл. 9.3. Кожному типу параметрів відповідає своя піктограма у вікні редактора реєстру.

Таблиця 9.3

Типи параметрів системного реєстру

Типи параметрів	Опис
1	2
REG_BINARY	Двійкові дані. Більшість відомостей про апаратні компоненти зберігаються у вигляді двійкових даних і виводяться в редакторові реєстру в шістнадцятковому форматі
REG_DWORD	Дані, подані цілим числом (4 байти). Багато параметрів служб і драйверів пристроїв мають цей тип та відображаються у двійковому, шістнадцятковому або десятковому форматах

Закінчення табл. 9.3

1	2
REG_SZ	Текстовий рядок фіксованої довжини
REG_EXPAND_SZ	Рядок змінної довжини. Цей тип даних містить змінні, що обробляються програмою або службою
REG_MULTI_SZ	Багаторядковий текст. Цей тип, переважно, мають списки та інші записи у форматі, зручному для читання. Записи розділяють пропусками, комами або іншими символами
REG_DWORD_LITTLE_ENDIAN	32-розрядне число у форматі "гострокінцевий" – молодший байт зберігається першим у пам'яті. Еквівалент REG_DWORD
REG_DWORD_BIG_ENDIAN	32-розрядне число у форматі "тупокінцевий" – старший байт зберігається першим у пам'яті
REG_QWORD	64-розрядне число
REG_QWORD_LITTLE_ENDIAN	64-розрядне число у форматі "гострокінцевий". Еквівалент REG_QWORD
REG_NONE	Параметр не має певного типу даних
REG_LINK	Символічне посилання Unicode. Тільки для внутрішнього використання (деякі кореневі розділи є таким посиланням на інші підрозділи)
REG_RESOURCE_LIST	Список апаратних ресурсів. Використовується тільки в розділі HKLM\HARDWARE
REG_FULL_RESOURCE_DESCRIPTOR	Дескриптор (описувач) апаратного ресурсу. Застосовують тільки у HKLM\HARDWARE
REG_RESOURCE_REQUIREMENTS_LIST	Список необхідних апаратних ресурсів. Використовують тільки у HKLM\HARDWARE

Водночас типи параметрів REG_BINARY, REG_DWORD і REG_SZ є основними та найбільш часто використовуваними типами.

Зберігання реєстру

Елементи реєстру зберігаються у вигляді атомарної структури. Реєстр розподілено на складові частини, так звані вулики (hives), або кущі. Вулики зберігаються на диску у вигляді файлів. Деякі вулики, такі як HKLM\HARDWARE, не зберігаються у файлах, а створюються за кожного

завантаження, тобто є змінними. Під час запуску системи реєстр збирається з вуликів у єдину деревоподібну структуру з кореневими розділами. Вулики реєстру і їхнє місцеположення на диску наведено в табл. 9.4.

Таблиця 9.4

Вулики реєстру

Вулики	Розташування
HKLM\SYSTEM	%SystemRoot%\system32\config\system
HKLM\SAM	%SystemRoot%\system32\config\SAM
HKLM\SECURITY	%SystemRoot%\system32\config\SECURITY
HKLM\SOFTWARE	%SystemRoot%\system32\config\software
HKLM\HARDWARE	Змінний вулик
HKLM\SYSTEM\Clone	Змінний вулик
HKU\<SID_користувача>	%USERPROFILE%\ntuser.dat
HKU\<SID_користувача>_Classes	%USERPROFILE%\Local ettings\Application Data\Microsoft\Windows\UsrClass.dat
HKU\.DEFAULT	%SystemRoot%\system32\config\default

де HKCU = HKEY_CURRENT_USER
HKLM = HKEY_LOCAL_MACHINE
HKU = HKEY_USERS
HKCC = HKEY_CURRENT_CONFIG
HKCR = HKEY_CLASSES_ROOT

Окрім цих файлів, є ряд допоміжних із такими розширеннями:

ALT – резервна копія вулика HKLM\SYSTEM (відсутній у XP);

LOG – журнал транзакцій, у якому реєструють усі зміни реєстру;

SAV – копії вуликів у тому вигляді, у якому вони були після завершення текстової фази встановлення.

Можливість обмеження доступу користувачів до того або того налаштування Windows здійснюють за допомогою зміни параметрів реєстру, що перебуває в розділі HKCU\SOFTWARE\Microsoft\Windows\Current Version \Policies***. Параметри мають тип DWORD і можуть набувати таких значень:

"1" – обмеження включене;

"0" – обмеження виключене.

Деякі параметри мають бути двійковими. Якщо необхідний такий параметр, то в таблиці на це буде вказано додатково.

Якщо деякі розділи, позначені зірочками, відсутні, то їх потрібно створити (табл. 9.5 – 9.12).

Таблиця 9.5

Обмеження доступу до налаштувань системи. *=System**

Параметри	Опис
NoDevMgrPage	Ховає "Пристрої" у властивостях "Мого комп'ютера"
NoConfigPage	Ховає "Профілі обладнання" у властивостях "Мого комп'ютера"
NoVirtMemPage	Ховає кнопку "Віртуальна пам'ять" на вкладці "Швидкодія"
NoFileSysPage	Ховає кнопку "Файлова система" на вкладці "Швидкодія"
DisableRegistryTools	Забороняє запуск regedit.exe

Таблиця 9.6

Обмеження доступу до налаштувань екрана. *=System**

Параметри	Опис
NoDispCPL	Робить недоступним діалог "Властивості: Екран"
NoDispBackgroundPage	Робить недоступною вкладку "Фон" діалогу "Властивості: Екран"
NoDispScrSavPage	Робить недоступною вкладку "Заставка"
NoDispAppearancePage	Робить недоступним "Оформлення"
NoDispSettingsPage	Робить недоступною вкладку "Налаштування"

Таблиця 9.7

Обмеження доступу до налаштувань паролів. *=System**

Параметри	Опис
NoSecCPL	Ховає "Паролі" в "Панелі управління"
NoPwdPage	Ховає кнопку "Зміна паролів" у властивостях "Паролі" "Панелі управління"
NoAdminPage	Ховає вкладку "Віддалене адміністрування" у властивостях "Паролі" "Панелі управління"
NoProfilePage	Ховає вкладку "Профілі користувачів" у властивостях "Паролі" "Панелі управління"

Таблиця 9.8

Обмеження доступу до налаштувань принтерів. ***=Explorer

Параметри	Опис
NoDeletePrinter	Забороняє видаляти принтер
NoAddPrinter	Забороняє додавати принтер
NoPrinterTabs	Забороняє доступ до "Властивостей принтера"

Таблиця 9.9

Обмеження доступу до налаштувань мережі. ***=Network

Параметри	Опис
NoNetSetup	Забороняє доступ до "Властивостей мережі" (із повідомленням)
NoNetSetupConfigPage	Забороняє доступ до "Властивостей мережі" (без повідомлення)
NoNetSetupIDPage	Недоступна вкладка "Ідентифікація" діалогу "Мережа"
NoNetSetupSecurityPage	Недоступна вкладка "Управління доступом" діалогу "Мережа"
NoEntireNetwork	Недоступна "Уся мережа" в "Мережевому оточенні"
NoFileSharingControl	Недоступно "відкриття" файлів і папок
NoPrintSharingControl	Недоступно "доступу" до принтерів
NoWorkgroupContents	Комп'ютери цієї робочої групи в "Мережевому оточенні" не відображаються

Таблиця 9.10

Обмеження можливостей меню "Пуск" і "Робочого столу". ***=Explorer

Параметри	Опис
1	2
NoRun	Ховає "Виконати" в меню "Пуск"
NoClose	Ховає "Завершення роботи" в меню "Пуск"
NoFind	Ховає "Знайти" в меню "Пуск"
NoLogOff	Значення двійкового параметра 01 00 00 00 ховає "Завершення сеансу" в меню "Пуск"
NoFavoritesMenu	Значення двійкового параметра 01 00 00 00 ховає "Виконати" в меню "Пуск"

Закінчення табл. 9.10

1	2
NoRecentDocsMenu	Значення двійкового параметра 01 00 00 00 ховає "Документи" в меню "Пуск"
NoSetTaskbar	Відключає "Налаштування/Панель завдань" у меню "Пуск"
NoSetFolder	Відключає "Налаштування" в меню "Пуск"
NoChangeStartMenu	Забороняє контекстне меню кнопки "Пуск"
NoNetHood	Прибирає значок "Мережеве оточення" з "Робочого столу"
NoDeskTop	Чистий "Робочий стіл"

Таблиця 9.11

Різне. *=Explorer**

Параметри	Опис
NoTrayContextMenu	Забороняє контекстне меню "Панелі завдань"
NoViewContextMenu	Забороняє контекстне меню "Мого комп'ютера"
NoSetActiveDesktop	Видаляє "Робочий стіл Active Desktop" із меню "Налаштування"
NoSaveSettings	Не запам'ятовувати налаштування під час виходу з Windows
NoFolderOption	Видаляє "Властивості папки" з меню "Налаштування"
NoRecentDocsHistory	Не запам'ятовувати недавно відкритих документів
ClearRecentDocsOnExit	Очистити список недавно відкритих документів під час виходу

Таблиця 9.12

Обмеження режиму MS-DOS. *=WinOldApp**

NoRealMode	Заборона перезавантаження в режимі емуляції MS-DOS
Disabled	Заборона запуску сеансу MS-DOS

Редактор реєстру

Редактор реєстру (regedit.exe) – це основний інструмент користувача для маніпуляції з реєстром, який надає Microsoft.

Для запуску редактора реєстру використовують меню "Пуск → Виконати". У вікні імені файла необхідно ввести regedit.exe і натиснути ОК (рис. 9.1). Редактор реєстру дає можливість переглядати вміст реєстру,

створювати та видаляти підрозділи і пари *параметр – значення*, виконувати експорт-імпорт усього реєстру або його частини.

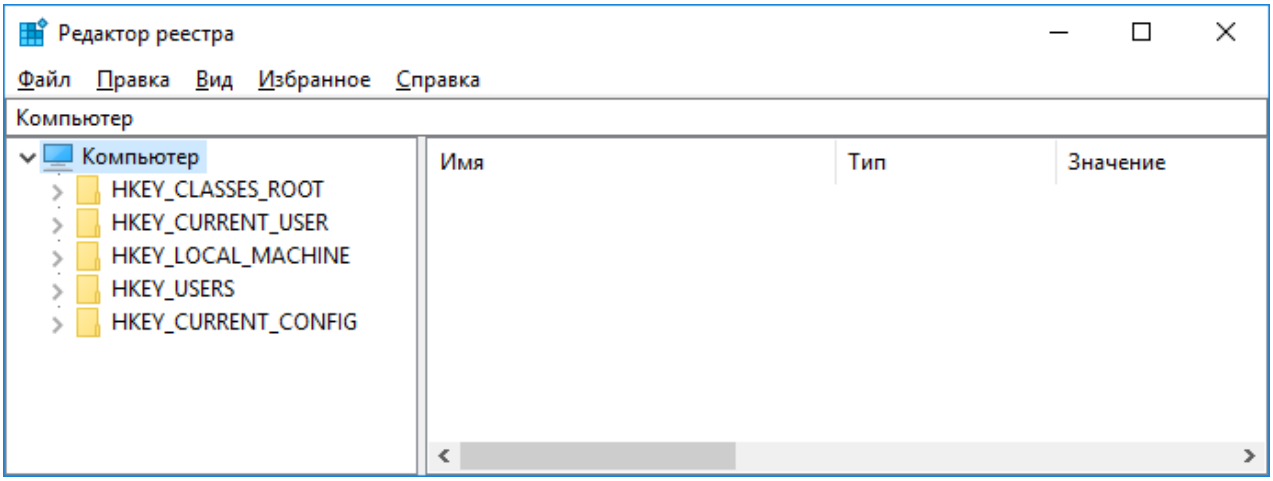


Рис. 9.1. **Головне вікно редактора реєстру**

Reg-файл – це файл, що має певну структуру і містить інформацію, яку може бути імпортовано в реєстр. Якщо було заблоковано роботу з редактором реєстру, то найбільш легким способом редагувати реєстр буде створення та імпортування reg-файла.

Під час побудови reg-файла можна скористатися блокнотом. На першому рядку мають розміщувати назву і версію редактора реєстру:

```
Windows Registry Editor Version 5.00
```

Після чого обов'язково слідує один порожній рядок.

Потім указано розділ реєстру, у якому треба прописати або змінити якісь параметри. Назву розділу має бути поміщено у квадратні дужки [...]. Далі прописують параметри, які треба додати, по одному параметру в рядку. Якщо треба зробити зміни в декількох розділах, то мають залишати один порожній рядок між останнім параметром попереднього розділу і назвою наступного розділу. Наприклад:

```
Windows Registry Editor Version 5.00

[Razdel1]
"param1"="znachenie1"
"param2"="znachenei2"
"param3"="znachenie3"
[Razdel2]
"param_1"="znachenie_1"
```

Останній рядок у файлі має бути **порожнім**. Після того як створено такий файл, необхідно просто запустити його як звичайну програму. Буде видано запит про необхідність зробити зміни в реєстрі, і після позитивної відповіді інформацію з файла буде імпортовано. Про результати імпортування Windows повідомить у вікні, що з'явилося після цього.

Тепер декілька слів про параметри, які можна додавати. У наведеному прикладі додають параметри за допомогою рядків типу "param1"="znachenie1". Тобто таким чином додається **рядковий** параметр з ім'ям "param1" і значенням "znachenie1". Але ж є ще й параметри двійкові та DWORD. Формат запису для їхнього додавання дещо інший. Для параметрів типу DWORD використовують такий рядок:

```
"param"=dword:XXXXXXXX
```

Тут "param" – ім'я параметра, dword – указує на тип цього параметра (букви мають бути обов'язково малі!) і після двокрапки слідує значення з восьми цифр у шістнадцятковому (!) форматі. Проте більшість параметрів DWORD мають значення або 0, або 1, значить, потрібно написати, відповідно, або 00000000, або 00000001, замість значків XXXXXXXX. Пропусків у рядку не допускають.

Для додавання двійкового параметра формат запису дещо інший:

```
"param"=hex:XX,XX,XX....
```

Тепер слід розшифрувати цей рядок. Із назвою параметра все ясно, після знака "=" слідує hex, тобто вказано, що це буде двійковий параметр, потім ідуть шістнадцяткові числа, відокремлені комою. Наприклад, якщо треба додати двійковий параметр, що дорівнює "be 00 00 00", то пишуть такий рядок:

```
"param"=hex:be,00,00,00
```

У реєстрі є параметри "За замовчуванням" (Default). Щоб надати їм якесь значення через reg-файл, треба додати такий рядок:

```
@="znachenie"
```

Тут значок @ показує, що надано значення параметра "За замовчуванням". Зверніть увагу на те, що його не беруть у лапки.

За допомогою reg-файла можна не тільки встановлювати нові параметри, але й видаляти їх. Наприклад, для видалення розділу з реєстру треба перед його ім'ям у квадратних дужках поставити символ "–". Ось який це має вигляд:

```
[-HKEY_LOCAL_MACHINE\Software\QuickSoft\QuickStart]
```

Для видалення окремих параметрів використовують такий синтаксис:

```
"param"= -
```

Редактор реєстру підтримує тільки основні типи даних параметрів реєстру. Для виконання операцій із даними інших типів необхідно використовувати функції Win32 API.

Функції Win32 для роботи з реєстром

У табл. 9.13 наведено всі функції системного реєстру, а опис основних слідує безпосередньо після цієї таблиці.

Таблиця 9.13

Зведення функцій роботи із системним реєстром

Функції	Призначення
1	2
RegCloseKey	Закриває відкритий ключ системного реєстру
RegConnectRegistry	Виконує з'єднання із зумовленим дескриптором системного реєстру на іншому комп'ютері
RegCreateKeyEx	Створює новий підключ
RegDeleteKey	Видаляє ключ із системного реєстру
RegDeleteValue	Видаляє значення із системного реєстру
RegEnumKeyEx	Перелічує всі підключі цього ключа
RegEnumValue	Перелічує всі значення цього ключа
RegFlushKey	Відразу ж записує всі зміни, зроблені в системному реєстрі
RegLoadKey	Завантажує розділ у кореневий ключ, що перебуває на вершині ієрархії
RegNotifyChangeKeyValue	Указує на момент зміни ключа або значення в системному реєстрі
RegOpenCurrentUser	Відкриває ключ HKCU для користувача поточного потоку
RegOpenKeyEx	Відкриває наявний ключ системного реєстру з розширенням Win32
RegOverridePredefKey	Перевизначає перевизначений ключ системного реєстру, відповідно до вказаного ключа системного реєстру
RegQueryInfoKey	Повертає інформацію про ключ
RegQueryMultipleValues	Вибирає тип і дані для списку імен значень
RegQueryValueEx	Повертає значення (із розширеними типами даних Win32)
RegReplaceKey	Замінює ключ умістом файла під час перезапуску системи

Закінчення табл. 9.13

1	2
RegRestoreKey	Прочитує вміст розділу в раніше збереженому ключі
RegSaveKey	Зберігає значення і підключі цього ключа у файлі вулика
RegSetValueEx	Надає ключу значення (із новими типами даних)
RegUnLoadKey	Видаляє розділ із системного реєстру

Програми для роботи з реєстром

До програм сторонніх виробників для роботи з реєстром належать такі:

1. *AccessEnum* (by Bryce Cogswell).
2. *AutoRuns* (by Mark Russinovich).
3. *Process Monitor* (by Mark Russinovich).
4. *RegShot* (freeware).
5. *Regcleaner* (by Jouni Vuorio).

AccessEnum. Корисна утиліта, що показує призначені права певним папкам або ключам реєстру. Застосування цієї програми дозволить визначити незахищені місця в системі й можливі напрями атак на неї (рис. 9.2).

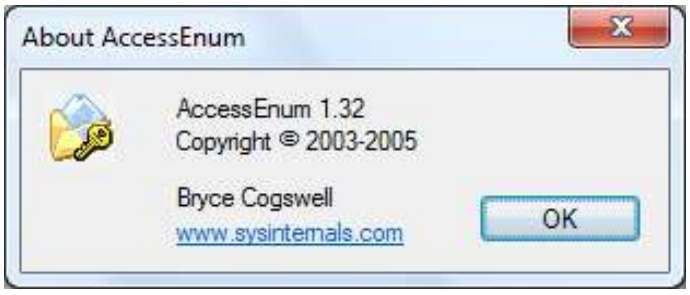


Рис. 9.2. **Діалогове вікно про програму AccessEnum**

Гнучка модель безпеки операційних систем Windows на основі технології NT дозволяє повною мірою управляти безпекою та правами доступу до файлів. Проте розподілити так, щоб усі користувачі мали належний доступ до файлів, каталогів і розділів реєстру, може виявитися складним завданням. *Убудованого в систему способу швидкого перегляду прав доступу користувачів до дерева каталогів або розділів реєстру не має.*

Програма AccessEnum створює детальний звіт за параметрами безпеки файлової системи й реєстру, що робить її ідеальним інструментом для пошуку вразливостей у системі безпеки та визначення зайвих прав доступу, які слід понизити.

Утиліта AccessEnum виконує сканування всіх файлів у вибраній теці або ключів у розділі реєстру і виводить результат у вигляді, зручному для перегляду користувачем (рис. 9.3).

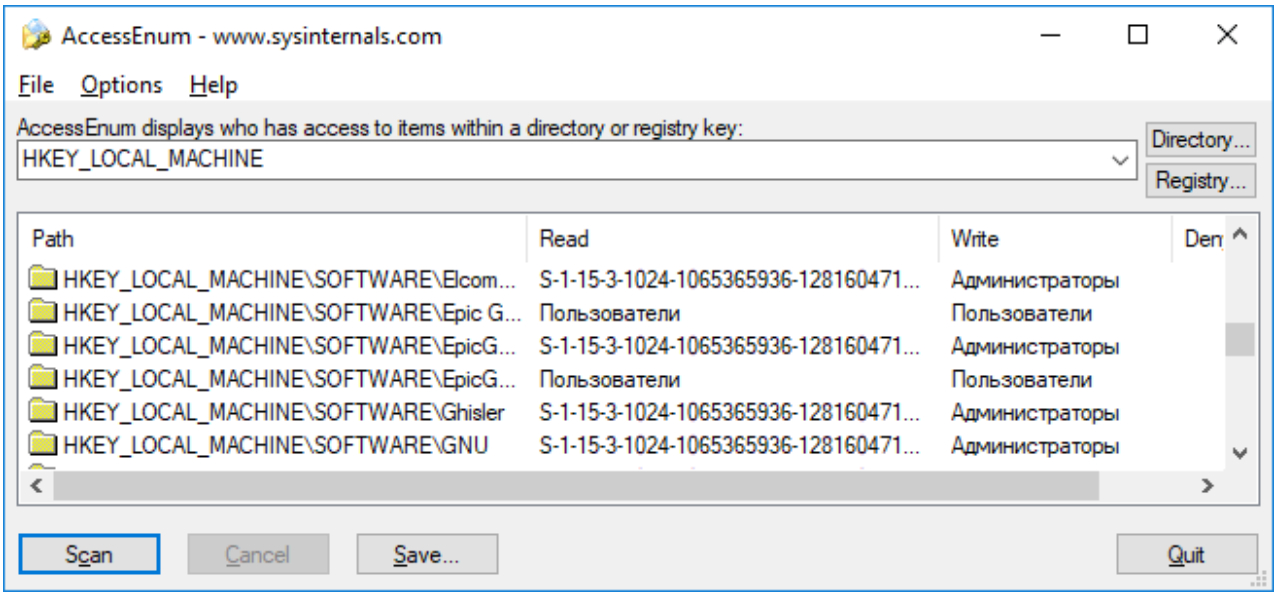
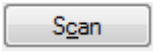


Рис. 9.3. **Головне вікно програми AccessEnum**

Інтерфейс програми простий, роботу не утруднено відсутністю русифікації. Після вибору папки або ключа реєстру варто натиснути кнопку 

. Визначену таблицю результатів завжди можна впорядковувати за ім'ям об'єкта, дозволеними або забороненими правами.

AutoRuns. Цей засіб перевіряє більшу кількість місць реєстру, із яких походить автозапуск програм, ніж будь-який інший монітор автозавантаження. Він показує, які програми налаштовано на запуск у процесі завантаження або входу в систему, причому ці програми відображаються в тому порядку, у якому система Windows їх обробляє.

Після запуску Autoruns, програма показує, які додатки налаштовано на автоматичний запуск, а також надає повний список розділів реєстру і каталогів файлової системи, які можуть використовуватися для задавання автоматичного запуску (рис. 9.4).

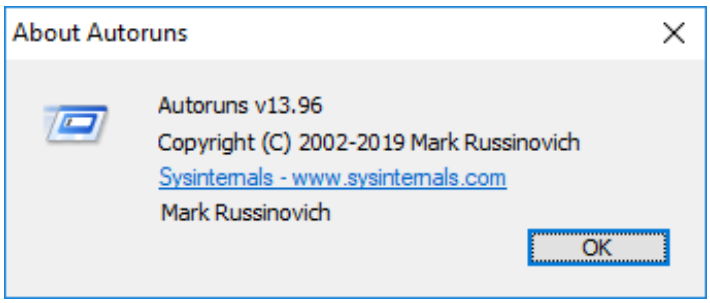


Рис. 9.4. **Діалогове вікно про програму AutoRuns**

Елементи, які показує програма Autoruns, належать до декількох категорій (рис.9.5):

об'єкти, що автоматично запускаються під час входу в систему;

додаткові компоненти провідника;

додаткові компоненти Internet Explorer (зокрема об'єкти модулів підтримки оглядача);

бібліотеки DLL-ініціалізації додатків, підміни елементів;

об'єкти, що виконуються на ранніх стадіях завантаження;

бібліотеки DLL-повідомлень Winlogon;

служби Windows і багаторівневі постачальники послуг Winsock.

Щоб переглянути об'єкти необхідної категорії, що автоматично запускаються, досить вибрати потрібну вкладку.

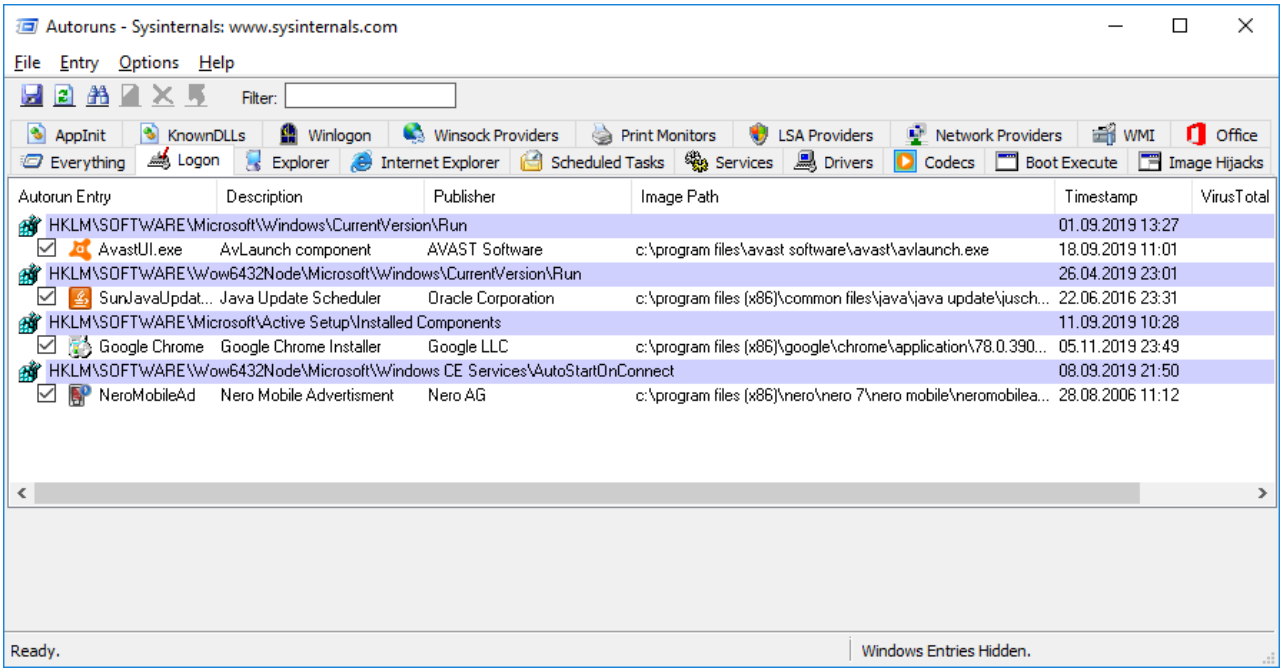


Рис. 9.5. **Головне вікно програми AutoRuns**

Щоб перейти до розділу реєстру або каталогу файлової системи, або до налаштування об'єкта, що запускається автоматично, досить виділити потрібний елемент і скористатися командою меню або кнопкою панелі інструментів **Jump** ("Перейти").

Щоб вимкнути об'єкт, що запускається автоматично, потрібно зняти відповідний йому прапорець. Видалити такий об'єкт можна за допомогою команди меню або кнопки панелі інструментів **Delete** ("Видалити").

Щоб переглянути елементи, що автоматично запускаються, для облікових записів інших користувачів, досить вибрати потрібний пункт меню **User** ("Користувач").

Process Monitor є вдосконаленим інструментом відстежування процесів для Windows, який у режимі реального часу відображає активність файлової системи, реєстру, а також процесів і потоків (рис. 9.6).

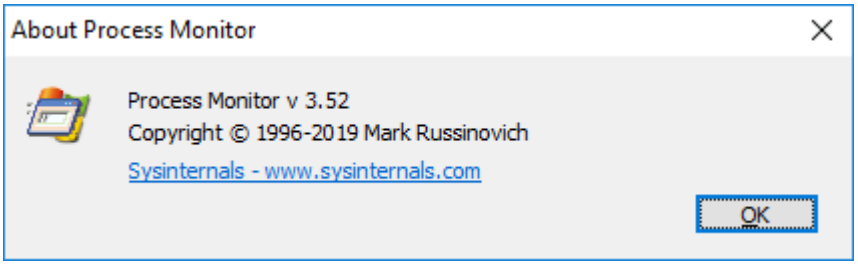


Рис. 9.6. **Діалогове вікно про програму Process Monitor**

У цій програмі поєднано такі можливості:

відстежування запуску та завершення роботи процесів і потоків, зокрема інформацію про код завершення;

відстежування завантаження образів (бібліотек DLL і драйверів пристроїв, що працюють у режимі ядра);

нешкідливі фільтри дозволяють установлювати фільтри, які не будуть призводити до втрати даних;

збирання стеків потоків для кожної операції дозволяє здебільшого визначити початкову причину виконання операції;

достовірне збирання інформації про процеси, зокрема шлях до образу процесу, командний рядок, а також ID-користувача та сесії;

колонки, що налаштовують і переміщують для кожної властивості події;

фільтри можна встановити на будь-яке поле з даними, зокрема поля, які не є колонками;

удосконалена архітектура запису журналів розширює можливості програми до десятків мільйонів зареєстрованих подій і гігабайтів записаних даних про події;

дерево процесів відображає відносини між всіма процесами, переліченими у відомостях трасування;

основний формат журналу зберігає всі дані, щоб їх можна було завантажити в іншому екземплярі програми Process Monitor;

підказування до процесів для простого перегляду інформації про образ процесу;

детальні підказки дозволяють дістати зручний доступ до форматів даних, які не поміщаються в колонці;

запис у журнал усіх операцій під час завантаження системи.

Найпростіше ознайомитися з можливостями програми Process Monitor, прочитавши файл довідки та спробувавши скористатися кожним пунктом меню і налаштування програми на робочій системі (рис. 9.7).

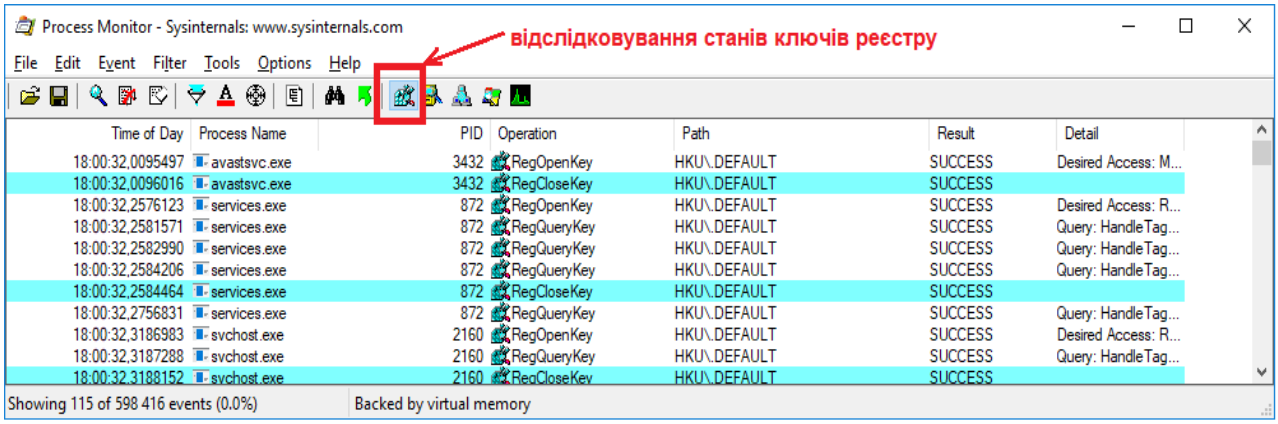


Рис. 9.7. **Головне вікно програми Process Monitor**

Щодо аналізу ключів реєстру програма Process Monitor надає таку унікальну можливість, як подання сумарної динамічної інформації за певний проміжок часу у вигляді таблиці.

Якщо необхідно відстежити події доступу до тільки конкретної гілки досить набудувати фільтр, указавши як шлях відповідний ключ. Наприклад, для розділу реєстру HKEY_Classes_Root за певний проміжок часу буде виконано 8 281 операцію, із яких (рис. 9.8):

144 операції доводиться на відкриття ключа;

147 – на закриття ключа;

7 988 – на операцію читання;

0 – на операцію запису параметрів (значень параметрів);

2 – на інші операції.

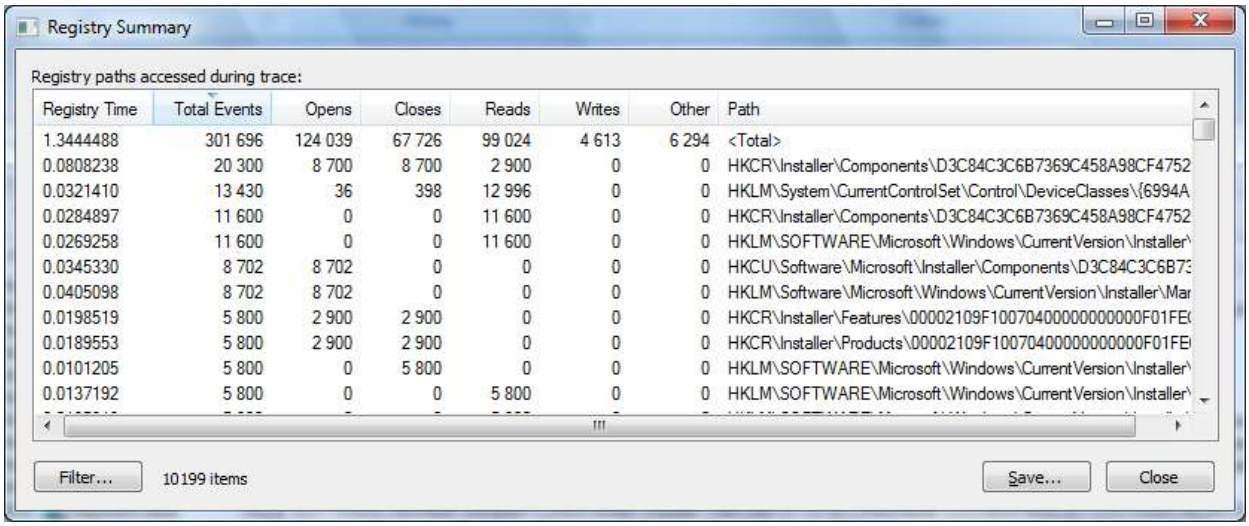


Рис. 9.8. **Відстеження подій із ключами всього реєстру за певний проміжок часу**

Таким чином, установлюючи відповідне значення ключа у фільтр, можна акцентовано відстежувати операції тільки із цим ключем (рис. 9.9).

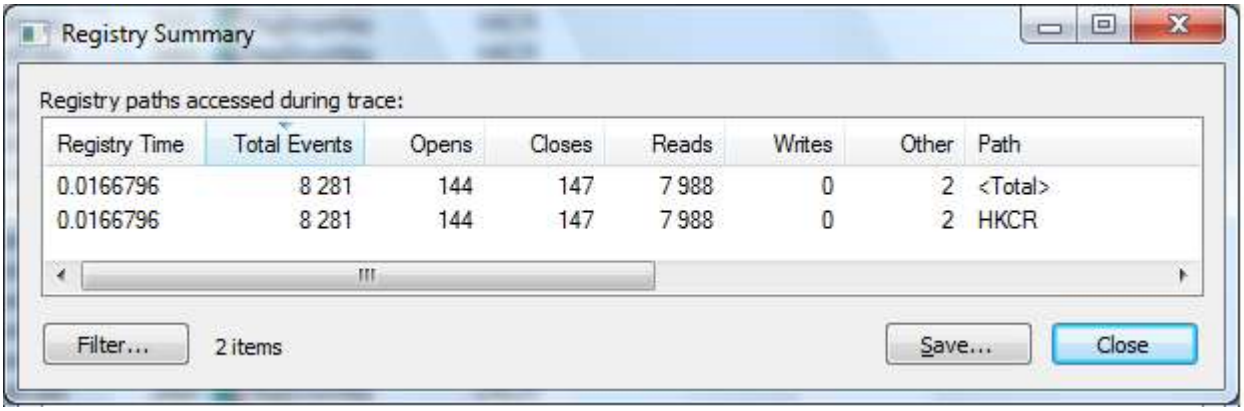


Рис. 9.9. **Відстеження подій тільки з ключем HKEY_Classes_Root**

За кожною подією доступу до реєстру додатково можна отримати **загальну** інформацію, до якої належить:

системний час, коли було здійснено доступ (Time of Day);

ім'я процесу, який здійснив доступ до ключа реєстру (Process Name);

ідентифікатор процесу (PID);

назва функції API, яку застосовано для виконання дії (Operation);

назва ключа (Path).

Також можна отримати **детальну** інформацію у вікні властивостей (Ctrl+P):

ідентифікатор потоку (Thread);

результат операції ("Успіх", "Відмова", "Ім'я не знайдено" тощо);

тривалість операції (Duration).

RegShot. Утиліту призначено для фіксації змін у реєстрі Windows. Вона може робити знімок реєстру, зберігати його у файлі, завантажувати з файла, порівнювати два знімки, знаходити всі відмінності (що змінилося, що було видалено, що з'явилося нового) (рис. 9.10).

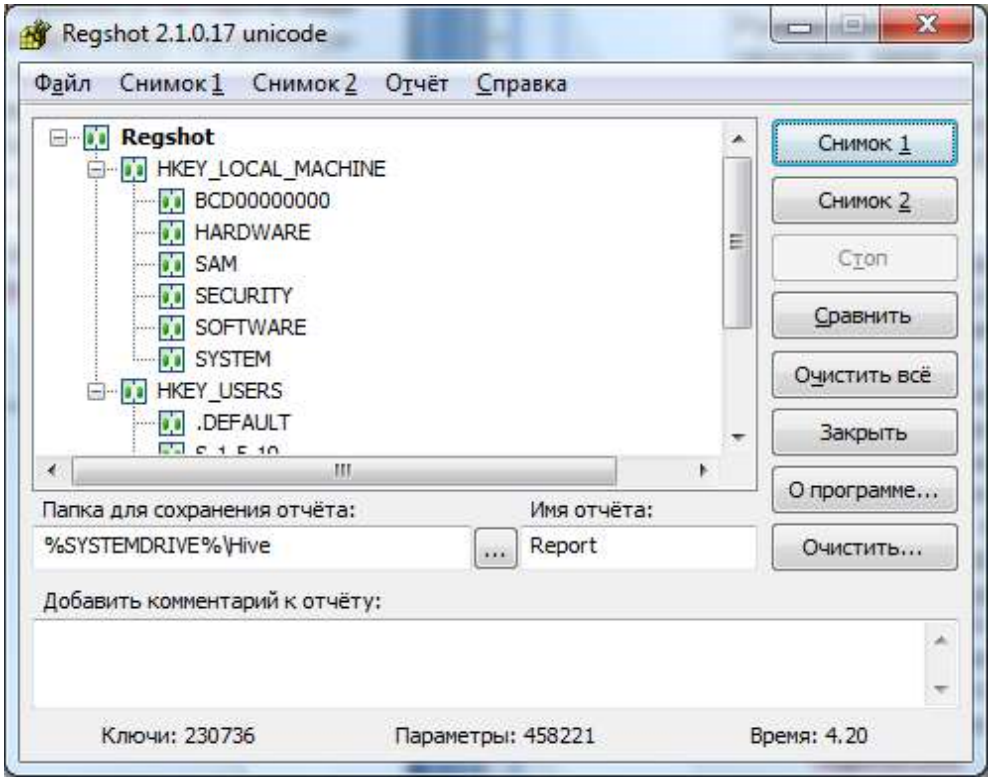


Рис. 9.10. **Головне вікно програми Process Monitor**

За наслідками змін формуються декілька звітів. HTML-звіт містить у наочному вигляді характеристики знімків і список усіх змін. Два звіти у форматі редактора реєстру undo.reg і redo.reg містять необхідну інформацію для приведення реєстру у стан, відповідний більш ранньому/пізньому знімку, відповідно. Два звіти у форматі настановних файлів undo.inf

і redo.inf містять аналогічну інформацію з тією самою метою. Кількістю формованих звітів можна управляти за допомогою налаштувань. Утиліта не інтегрується в систему, не записує в реєстр інформацію, усі її налаштування зберігаються у файлі regshot.xml.

RegShot може стати в нагоді, якщо система почне нестабільно працювати та знадобиться з'ясувати, які зміни в реєстрі призвели до цієї нестабільності.

Також цей спосіб може бути застосовано для продовження працездатності shareware-програм, які працюють обмежений період часу, наприклад 30 днів, а після просять реєстрації.

Далі наведено приклад звітного звіту, що формується програмою RegShot (рис. 9.11).

Звітний звіт		
Параметри	Знімок A	Знімок B
Дата знімання	09.04.2019 18:40:36	09.04.2019 18:41:52
Комп'ютер	AAAA	AAAA
Користувач	BB	BB
SID користувача	S-1-5-21-22051-214127-1179179250-1000	S-1-5-21-22051-214127-1179179250-1000
Тип знімання	Реєстр повністю	Реєстр повністю
Час знімання	4.27	4.20
Помилки	600	600
Ключі	230735	230736
Видалені/нові ключі	0	1
Змінені ключі (дозволи)	0	0
Параметри	458220	458221
Видалені/нові параметри	0	1
Змінені параметри	5	5
Усього змін	5	7
Файл відновлення реєстру .reg	Report.2.UndoReg.txt	Report.2.RedoReg.txt

Рис. 9.11. **Приклад звіту, що формується програмою RegShot**

Regcleaner – це версія програми очищення реєстру Windows. За допомогою цієї програми можна підвищити продуктивність системи, звільнивши

її реєстр від залишків некоректно видаленого програмного забезпечення (рис. 9.12).

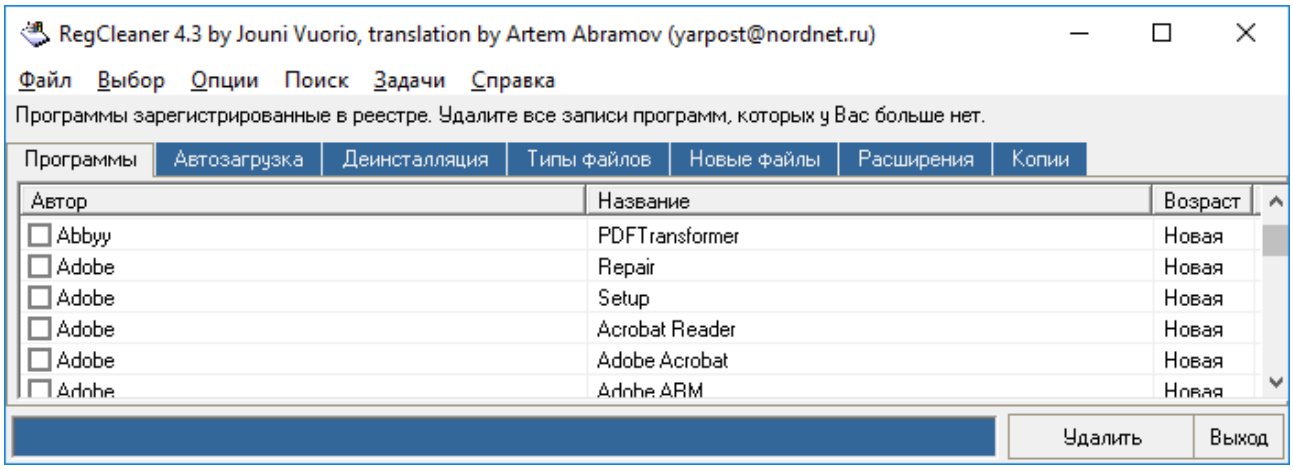


Рис. 9.12. **Головне вікно програми Regcleaner**

RegCleaner зручний тим, що не дає можливості користувачеві внести зміни, що призводять до втрати працездатності системи або появи помилок, чого рідко вдається уникнути під час роботи з незрозумілою для більшості користувачів структурою запису даних у реєстрі.

Програма, яка дозволяє видаляти застарілі записи в реєстрі Windows (наприклад, під час видалення програмного забезпечення, можна знайти залишки програми). У інтерфейсі **RegCleaner** перебуває сім укладок: програми, автозавантаження, деінсталяція, типи файлів, нові файли, розширення, а також резервне копіювання. У кожній колонці можна побачити автора і назву програми, шлях до файла, джерело, розширення й опис, команди, а також дату та час резервних копій.

Переваги:

ретельний аналіз реєстру;

структуру інтерфейсу виконано у стилі, зручному користувачу;

можливість вибору ручного або автоматичного чищення реєстру;

зручна система виділення;

інформативні підказування у верхній частині програми;

можливість створення резервної копії реєстру перед чищенням;

складний і ефективний алгоритм, який програма використовує для пошуку й очищення застарілих записів;

сканування реєстру на наявність невживаних DLL-файлів;

підтримка російської мови в інтерфейсі, що набагато спрощує роботу;

нескладні налаштування в інтерфейсі;

розділ управління автозавантаженням;

після чищення реєстру значно зменшується час завантаження операційної системи та власне помітно підвищується продуктивність.

Недоліки:

у ручному режимі очищення не зовсім зрозуміло, що можна видаляти, а що ні;

за заявою виробника програма видаляє тільки непотрібні файли, але бувають випадки, коли після повного очищення деякі програми не відкривалися;

виявлено проблеми в роботі програми на багатоядерних процесорах.

Загальне завдання для виконання

Увага!

1. Перед початком редагування реєстру обов'язково підготуйте копію відповідної гілки реєстру.

2. Ніколи не видаляйте підрозділи та пари *параметр – значення,* створені не вами!

Етап 1. Кожен студент, відповідно до свого індивідуального варіанта (п. 1), має провести дослідження певної гілки системного реєстру. У результаті дослідження він заповнює табл. 9.14.

Таблиця 9.14

Дослідження гілки системного реєстру

№ п/п	Ключі (підключі)	Параметри	Значення	Пояснення
1	HCC\Software\Fonts			
…	…	…	…	…

Етап 2. Проведіть дослідження прав доступу до ключів розділу реєстру, відповідно до свого індивідуального варіанта (п. 1), за допомогою програми AccessEnum. Результат наведіть у вигляді табл. 9.15.

Таблиця 9.15

Дослідження прав доступу до гілки системного реєстру

Шлях до гілки реєстру	Права на читання	Права на запис	Права на відмову в доступі	Пояснення

Етап 3. Проведіть дослідження розділу HKEY_Classes_Root у Windows 7 щодо асоційованих додатків, прописаних у системі стандартним розширенням імен файлів, відповідно до свого індивідуального варіанта (п. 2). Результат запишіть у табл. 9.16.

Таблиця 9.16

Дослідження асоційованих додатків у HKEY_Classes_Root

Розширення файла	Значення параметра за замовчуванням	Шлях до зіставленого додатка з параметрами
.txt	txtfile	%SystemRoot%\system32\ NOTEPAD.EXE %1

Етап 4. Проведіть дослідження заданих для атозапуску під час старту систем програми, відомості про яких зберігаються в системному реєстрі. Для цього використовуйте програму **AutoRuns** для Windows. Обов'язково розпишіть назви гілок реєстру, у яких прописано автоматично завантажувані файли. Результати занесіть до табл. 9.17.

Таблиця 9.17

Монітор автозавантаження

Назви	Опис	Виготівники	Шлях до файла	Дата створення файла

Етап 5. Проведіть дослідження доступу до ключа реєстру, вибраний довільно з розділу, указаного в п. 3 індивідуального завдання, у реальному масштабі часу за допомогою програми **Process Monitor**. Результати занесіть до табл. 9.18.

Таблиця 9.18

Дослідження ключа реєстру в реальному масштабі часу

Назва ключа	Ім'я процесу	PID	TID	Операція (функція)	Результат	Тривалість

Подайте скриншот узагальненої статистики доступу до цього ключа впродовж 5 хв роботи у вигляді табл. 9.19.

Таблиця 9.19

Дослідження узагальненої статистики доступу до ключа реєстру в реальному масштабі часу

Назва ключа	Усього подій	Відкриття ключа	Закриття ключа	Читання	Запис	Інша операція

Етап 6. Розробіть **2 reg-файли системного реєстру:**

1-й reg-файл – для переміщення в реєстрі гілки, указаної в п. 1 індивідуального завдання, даних про власне прізвище. Таким чином, прізвище має бути подано в реєстрі у вигляді 14 різних типів даних (див. табл. 9.3). Як числові типи використовуйте зображення прізвища студента в ASCII-кодах.

2-й UnReg-файл – для видалення внесених першим файлом ключів, параметрів і їхніх значень.

Етап 7. Проведіть дослідження зміни налаштування реєстру під час інсталяції shareware програмного забезпечення, заданого в п. 4 індивідуального завдання, використовуючи програму **RegShot**. До інсталяції

зніміть із реєстру знімок 1. Виконайте інсталяцію. Зніміть знімок 2. Зробіть їхнє порівняння. Відстежте гілки, ключі та параметри, які було додано (змінено) програмою, що інсталювалася.

У звіт помістіть відредагований звіт, який генерується програмою **RegShot.** Під редагуванням розуміють відображення тільки розділів, які містять **ненульові** значення.

Додаткове завдання

Розробіть додаток, що дозволяє управляти вказаними в п. 5 елементами Windows. Управління має дозволити користувачеві як установлювати, так і відміняти встановлення параметрів.

Індивідуальні завдання для виконання

Варіант № 1

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\PnP.
2. Розширення: .bmp; .js; .evt.
3. Назва розділу: HKEY_CLASSES_ROOT.
4. Shareware-додаток для інсталяції: memory-card-data-recovery-demo.exe.
5. Контекстне меню панелі завдань, меню папок і файлів. Гілка HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer HKCR*\shellex\ContextMenuHandlers, HKCR\Directory\shell, HKCR\Folder\shell Ключ: No Tray ContextMenu.

Варіант № 2

1. HKEY_CURRENT_USER\Control Panel\Mouse.
2. Розширення: .m1v; .der; .png.
3. Назва розділу: HKEY_CLASSES_ROOT\Installer.
4. Shareware-додаток для інсталяції: adkm.exe.
5. Утаєння значків дисків у вікні "Мій комп'ютер" і "Провідник" та заборона на доступ до вмісту вибраних дисків. Гілка HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer, параметри NoDrives і NoViewOnDrive.

Варіант № 3

1. HKEY_USERS\.DEFAULT\Control Panel\Colors.
2. Розширення: .aac; .mpe; .gadget.
3. Назва розділу: HKEY_CLASSES_ROOT\Setting.
4. Shareware-додаток для інсталяції: cdbfwgui.exe.
5. Управління можливостями меню "Пуск" (див. табл. 9.10).

Варіант № 4

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\IDE.
2. Розширення: .exc; .hlp; .ac3.
3. Назва розділу: HKEY_CURRENT_USER.
4. Shareware-додаток для інсталяції: MemopumpSetup.zip.
5. Діалогове вікно відкриття та збереження файла. Гілка HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\comdlg32, розділ "PlacesBar", параметри: NoPlacesBar, NoBackButton, NoFileMru.

Варіант № 5

1. HKEY_CURRENT_USER\Control Panel\Keyboard.
2. Розширення: .ofs; .camp; .jtx.
3. Назва розділу: HKEY_CURRENT_USER\System.
4. Shareware-додаток для інсталяції: lb404setup.exe.
5. Обмеження режиму MS-DOS (див. табл. 9.12).

Варіант № 6

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls].
2. Розширення: .3gp; .dtd; .mpa.
3. Назва розділу: HKEY_CURRENT_USER\Software.
4. Shareware-додаток для інсталяції: EXE-extractor.exe.
5. Диспетчер завдань Windows і Синій Екран Смерті Windows. Гілки HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System і HKLM\SYSTEM\CurrentControlSet\Servises\i8042prt\Parameters, відповідно. Ключі: DisableTaskMgr і CrashOnCtrlScroll.

Варіант № 7

1. HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System.
2. Розширення: .ps1; .crt; .jpeg.
3. Назва розділу: HKEY_CURRENT_USER\Control Panel.

4. Shareware-додаток для інсталяції: mnkPlus.exe.

5. Доступ до налаштувань мережі (див. табл. 9.9).

Варіант № 8

1. HKLM\SYSTEM\CurrentControlSet\Enum\STORAGE.

2. Розширення: .css; .pbk; .jpe.

3. Назва розділу: HKEY_LOCAL_MACHINE\Software.

4. Shareware-додаток для інсталяції: OE-Mail recovery.exe.

5. Управління годинником, яке містить: синхронізацію системного годинника, вибір time-серверів, прикрасу годинничків. Гілки: HKLM\SYSTEM\ControlSet001\Services\W32Time\TimeProviders\NtpClient, HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\DateTime\Servers, HKCU\Control Panel\International. Ключі: SpecialPollInterval, sTimeFormat.

Варіант № 9

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa.

2. Розширення: .gmmp; .au; .fdf.

3. Назва розділу: HKEY_LOCAL_MACHINE.

4. Shareware-додаток для інсталяції: MSItoEXECreatorDemo.exe.

5. Управління можливостями "Робочого столу" (див. табл. 9.10).

Варіант № 10

1. HKEY_USERS\.DEFAULT\Identities.

2. Розширення: .msu; .iso; .asx.

3. Назва розділу: HKEY_USERS.

4. Shareware-додаток для інсталяції: phones.exe.

5. Дискові операції з перевірки диска: автоматичному виправленню помилок, зміні часу очікування. Гілки: HKU\DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Applets\CheckDrive і HKLM\SYSTEM\CurrentControlSet\Control\SessionManager. Ключі: AutoChk і AutoChkTimeOut.

Варіант № 11

1. HKEY_CURRENT_CONFIG\Software\Fonts.

2. Розширення: .emf; .cda; .mov.

3. Назва розділу: HKEY_CLASSES_ROOT.

4. Shareware-додаток для інсталяції: PrvDisk.exe.

5. Повідомлення під час завантаження, автозавантаження. Гілки HKLM\Software\Microsoft\WindowsNT\CurrentVersion\Winlogon і HKLM\ Software\Microsoft\Windows\CurrentVersion. Ключі: LegalNoticeCaption, LegalNoticeText, RunOnce, RunOnceEx, RunServices, RunServicesOnce, RunServices, DisableLocalMachineRun, DisableLocalMachineRunOnce, DisableCurrentUserRun).

Варіант № 12

1. HKEY_CURRENT_USER\Printers.

2. Розширення: .icm; .rat; .docm.

3. Назва розділу: HKEY_CLASSES_ROOT\Installer.

4. Shareware-додаток для інсталяції: RecoveryToolboxForAccess Install.exe.

5. Заборона на доступ до вмісту вибраних дисків. Гілка HKCU\Software\ Microsoft\Windows\CurrentVersion\Policies\Explorer. Ключ: NoViewOnDrive. Організуйте діалог із вибору назв маскованих дисків.

Варіант № 13

1. HKCU\Software\Microsoft\Windows\CurrentVersion\GroupPolicy\Group-Membership.

2. Розширення: .cpl; .ifo; .nfo.

3. Назва розділу: HKEY_CLASSES_ROOT\Setting.

4. Shareware-додаток для інсталяції: setup_akl.exe.

5. Дискові операції, такі як зміна порогу видавання попередження про брак вільного місця на диску, дефрагментація. Гілки:

HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters;

HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer;

HKLM\System\CurrentControlSet\Control\FileSystem;

HKCU\Software\Microsoft\Windows\CurrentVersion\Applets\Defrag\Settings;

HKLM\SOFTWARE\Microsoft\Dfrg\BootOptimizeFunction;

HKLM\SYSTEM\CurrentControlSet\ Control\SessionManager\Memory Management,

HKLM\SYSTEM\CurrentControlSet\Control\SessionManager\Memory Management.

Ключі: DiskSpaceThreshold, NoLowDiskSpaceChecks, ContigFileAllocSize, DisableScreenSaver, Enable, ClearPageFileAtShutdown, DisablePagingExecutive.

Варіант № 14

1. HKLM \SYSTEM\CurrentControlSet\Enum\DISPLAY.
2. Розширення: .reg; .adt; .dwfx.
3. Назва розділу: HKEY_CURRENT_USER.
4. Shareware-додаток для інсталяції: RecoveryToolboxForOutlook ExpressInstall.exe.
5. Видалення стрілки з ярличка, "долоньки" з відкритими ресурсами, збільшення швидкості спливання меню. Гілки HKCR\lnkfile, HKEY_CLASSES_ROOT\piffile, HKCR\Network\SharingHandler, HKEY_CURRENT_USER\ControlPanel\Desktop. Параметри: IsShortcut, за умовчанням, Menu ShowDelay.

Варіант № 15

1. HKLM \SYSTEM\CurrentControlSet\Control\Print\ Printers.
2. Розширення: .hta; .pfx; .dqy.
3. Назва розділу: HKEY_CURRENT_USER\System.
4. Shareware-додаток для інсталяції: ScreenshotMakerPro_Setup.exe.
5. Заборона запуску програм, редактора реєстру. Гілки HKEY_CURRENT_USER\SOFTWARE\Microsoft\ Windows\CurrentVerson\Policies\ Explorer та HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\Current Verson\Policies\System. Ключі: RestrictRun і DisableRegistryTools, відповідно.

Варіант № 16

1. HKEY_USERS\.DEFAULT\Control Panel\International.
2. Розширення: .wab; .gif; .csv.
3. Назва розділу: HKEY_CURRENT_USER\Software.
4. Shareware-додаток для інсталяції: sim-card.exe.
5. Доступ до налаштування екрана (див. табл. 9.6).

Варіант № 17

1. HKEY_USERS\.DEFAULT\Control Panel\Mouse.
2. Розширення: .log; .flv; .cer.
3. Назва розділу: HKEY_LOCAL_MACHINE.
4. Shareware-додаток для інсталяції: swf-to-gif.exe.
5. Доступ до налаштувань системи (див. табл. 9.5).

Варіант № 18

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print\Monitors.

2. Розширення: .dic; .inf; .msp.

3. Назва розділу: HKEY_LOCAL_MACHINE\Software.

4. Shareware-додаток для інсталяції: vtrain-en50.exe.

5. Доступ до налаштувань принтерів (див. табл. 9.8).

Варіант № 19

1. HKEY_LOCAL_MACHINE\HARDWARE\ACPI.

2. Розширення: .cat; .htm; .msc.

3. Назва розділу: HKEY_CURRENT_USER\Control Panel.

4. Shareware-додаток для інсталяції: smart-pdf-converter-pro-setup.exe.

5. Панель перемикача завдань. Гілка HKEY_CURRENT_USER\ControlPanel\Desktop. Ключі: CoolSwitch, CoolSwitchRows і CoolSwitchColumns.

Варіант № 20

1. HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer.

2. Розширення: .jpg; .chm; .jtp.

3. Назва розділу: HKEY_USERS.

4. Shareware-додаток для інсталяції: wma-converter.exe.

5. Доступу до налаштувань паролів (див. табл. 9.7).

Варіант № 21

1. HKEY_USERS\.DEFAULT\Control Panel\Desktop.

2. Розширення: .ape; .icc; .url.

3. Назва розділу: HKEY_CLASSES_ROOT.

4. Shareware-додаток для інсталяції: wzipse40.exe.

5. Екран вітання Windows. Гілка HKU\.DEFAULT\ControlPanel\Desktop, значення 2 для ключів: FontSmoothing, FontSmoothingType, і значення 1 для ключа FontSmoothingOrientation.

Варіант № 22

1. HKLM\HARDWARE\RESOURCEMAP\PnPManager\PnpManager.

2. Розширення: .rle; .dot; .jnt.

3. Назва розділу: HKEY_CLASSES_ROOT\Setting.

4. Shareware-додаток для інсталяції: lb404setup.exe.

5. Паролі та безпека. Гілка HKLM\SOFTWARE\Microsoft\Windows\ CurrentVersion\Policies\Network. Параметри: NoDialIn, DisablePwdCaching, HideSharePwds, NoFileSharing, NoFileSharingControl, NoPrintSharing, NoPrint SharingControl.

Варіант № 23

1. HKEY_CURRENT_USER\Environment.
2. Розширення: .tif; .ini; .divx.
3. Назва розділу: HKEY_CLASSES_ROOT\Installer.
4. Shareware-додаток для інсталяції: EXE-extractor.exe.
5. Командний рядок. Гілка HKLM\SYSTEM\CurrentControlSet\Control\ Session Manager\Environment. Параметр PROMPT типу REG_EXPAND_SZ із значеннями:

$B | (вертикальна межа)	$D поточна дата
$G > (знак більше)	$L < (знак менше)
$N поточний диск	$P поточний диск і шлях
$Q = (знак дорівнює)	$T поточний час
$V версія Windows	$$ $ (знак долара)

Варіант № 24

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\PnP.
2. Розширення: .crd; .svg; .odc.
3. Назва розділу: HKEY_CURRENT_USER\System.
4. Shareware-додаток для інсталяції: mnkPlus.exe.
5. Контекстне меню панелі завдань, меню папок і файлів. Гілки HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\ Explorer, HKCR\ *\shellex\ ContextMenuHandlers, HKCR\Directory\shell, HKCR\Folder\shell. Ключ NoTrayContextMenu.

Варіант № 25

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB.
2. Розширення: .sct; .diagcab; .img.
3. Назва розділу: HKEY_CURRENT_USER.
4. Shareware-додаток для інсталяції: memory-card-data-recovery-demo.exe.
5. Реєстраційні дані. Гілка HKLM\SOFTWARE\Microsoft\WindowsNT\ CurrentVersion.

ВАРІАНТ № 26

1. HKEY_CURRENT_USER\Control Panel\Mouse.
2. Розширення: .msi; .pdf; .crl.
3. Назва розділу: HKEY_CURRENT_USER\Software.
4. Shareware-додаток для інсталяції: MSItoEXECreatorDemo.exe.
5. Приховання значків дисків у вікні "Мій комп'ютер" і "Провідник" і заборона на доступ до вмісту вибраних дисків. Гілка HKCU\Software\ Microsoft\Windows\CurrentVersion\Policies\Explorer. Параметри: NoDrives і NoViewOnDrive.

ВАРІАНТ № 27

1. HKEY_USERS\.DEFAULT\Control Panel\Colors.
2. Розширення: .avi; .ico; .ppt.
3. Назва розділу: HKEY_CURRENT_USER\Control Panel.
4. Shareware-додаток для інсталяції: swf-to-gif.exe.
5. Управління можливостями меню "Пуск" (див. табл. 9.10).

Варіант № 28

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\IDE.
2. Розширення: .odt; .fon; .aif.
3. Назва розділу: HKEY_LOCAL_MACHINE.
4. Shareware-додаток для інсталяції: OE-Mail recovery.exe.
5. Діалогове вікно відкриття і збереження файла. Гілка HKCU\ SOFT WARE\Microsoft\Windows\CurrentVersion\Policies\comdlg32, розділ "PlacesBar", параметри: NoPlacesBar, NoBackButton, NoFileMru.

Варіант № 29

1. HKEY_CURRENT_USER\Control Panel\Keyboard.
2. Розширення: .doc; .iqy; .mod.
3. Назва розділу: HKEY_LOCAL_MACHINE\Software.
4. Shareware-додаток для інсталяції: phones.exe.
5. Обмеження режиму MS-DOS (див. табл. 9.12).

Варіант № 30

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls].
2. Розширення: .jse; .blg; .group.

3. Назва розділу: HKEY_USERS.

4. Shareware-додаток для інсталяції: PrvDisk.exe.

5. Диспетчер завдань Windows і Синій Екран Смерті Windows. Гілки HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System і HKLM\SYSTEM\CurrentControlSet\Servises\i8042prt\Parameters, відповідно. Ключі: DisableTaskMgr і CrashOnCtrlScroll.

Контрольні запитання

1. Із яких файлів складається реєстр? Де вони розташовані?
2. Поясніть структуру системного реєстру.
3. Назвіть архітектуру основного розділу HKEY_CLASSES_ROOT.
4. У чому полягає призначення основних розділів реєстру?
5. Що таке "параметр ключа"?
6. Назвіть типи параметрів і їхні можливі значення.
7. Укажіть способи відновлення реєстру.
8. У чому полягає специфіка гілок реєстру?
9. Із якою метою використовують розділ HKEY_CURRENT_USER?
10. Поясніть спосіб задавання в reg-файлі параметра REG_EXPAND_SZ.
11. Поясніть спосіб задавання в reg-файлі параметра REG_MULTI_SZ.
12. Поясніть спосіб задавання в reg-файлі параметра REG_DWORD_LITTLE_ENDIAN.
13. Поясніть спосіб задавання в reg-файлі параметра REG_DWORD_BIG_ENDIAN.
14. Поясніть спосіб задавання в reg-файлі параметра REG_QWORD_LITTLE_ENDIAN.
15. Поясніть спосіб задавання в reg-файлі параметра REG_RESOURCE_LIST.
16. Поясніть спосіб задавання в reg-файлі параметра REG_DWORD.
17. Поясніть спосіб задавання в reg-файлі параметра REG_FULL_RESOURCE_DESCRIPTOR.
18. Поясніть спосіб задавання в reg-файлі параметра REG_BINARY.
19. Назвіть самий вагомий розділ системного реєстру. Обґрунтуйте свою відповідь.
20. Який розділ системного реєстру залежить від поточної сесії?

Розділ 4. Мережеві багатопроцесорні операційні системи та захист інформації

Лабораторна робота 10
Дослідження системних служб і драйверів

Мета роботи: ознайомлення з одними з найбільш важливих структурних елементів Windows – системними службами та драйверами. Набуття практичних навичок у створенні служб, їхньої інсталяції та реєстрації в системному реєстрі операційної системи, а також досконально вивчити оточення, у якому виконуються системні служби та драйвери.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно вивчити порядок роботи Диспетчера системних служб та залежності між різними sys-файлами у Windows.

Додаткову інформацію під час підготовки до роботи можна отримати в таких джерелах [2; 21].

Теоретичні відомості

Системна служба Windows (по-англійськи – *system service*; надалі – просто служба) – це фоновий процес, який може запускатися і працювати без участі інтерактивного користувача.

Архітектура системних служб Windows має на увазі три види компонентів (рис. 10.1). Її ядром є *Диспетчер системних служб* – **Service Control Manager (SCM)**. API для "спілкування" із SCM експортує стандартна бібліотека Windows advapi32.dll. Підсистема служб завантажується на самому початку старту ОС із виконуваного файла services.exe, однойменний процес можна спостерігати у списку активних процесів увесь час роботи комп'ютера. Бібліотека advapi32.dll спілкується із services.exe за допомогою RPC (Remote Procedure Call).

Рис. 10.1. **Архітектура системних служб Windows**

SCM запускається під час завантаження системи та працює, поки комп'ютер не буде вимкнено. Диспетчер системних служб взаємодіє з одного боку із *керівними програмами*, а з іншого – із системними службами.

Відмінність служби від звичайної програми в тому, що служба, яка є запущеною, працює і після виходу користувача із системи навіть тоді, коли виводиться запрошення "Для входу в систему натисніть Ctrl+Alt+Delete". Тому у службі найчастіше оформляють такі програми, які мають надавати якісь послуги навіть тоді, коли не має користувачів, що увійшли до системи. Із вбудованих служб Windows це, наприклад:

Spooler (служба друку);

Alerter (служба сповіщення, часто використовувана системними адміністраторами);

Server (дозволяє відкривати файли, named pipes та інші комп'ютери іншими машинами в мережі) тощо.

Основними завданнями диспетчера SCM є:

запуск під час завантаження системи тих служб, які мають бути запущені автоматично;

зберігання конфігураційної бази даних, що містить інформацію про всі служби;

приймання запитів від керівних програм і передавання їх системним службам.

Windows визначає два види системних служб. Один із них – це так звані **служби режиму ядра** (kernel-mode services), які є нічим іншим, як драйверами пристроїв.

Інший вид – служби Win32 – це звичайні Win32-процеси, що використовують спеціальний набір функцій для взаємодії з Диспетчером системних служб.

Керівні програми – це звичайні Win32-додатки, які маніпулюють службами з використанням програмного інтерфейсу, що надається Диспетчером системних служб. Типовий приклад керівної програми – це mmc.exe.

Драйвери пристроїв є завантажуваними модулями режиму ядра (переважно, це файли з розширенням .sys); вони утворюють інтерфейс між диспетчером уведення-виведення і відповідним обладнанням. Ці драйвери виконують у режимі ядра в одному із трьох контекстів:

у контексті призначеного для користувача потоку, що ініціював функцію введення-виведення;

у контексті системного потоку режиму ядра;

як результат переривання (а значить, не в контексті якого-небудь процесу або потоку, який був поточним на момент переривання).

У Windows драйвери пристроїв не управляють обладнанням безпосередньо, замість цього вони викликають функції HAL. Драйвери, переважно, пишуть на C (іноді на C++). Тому в разі правильного використання процедур HAL вони є переносними між підтримуваною архітектурою Windows на рівні початкової коди, а на рівні двійкових файлів – усередині сім'ї з однаковою архітектурою.

Є декілька типів драйверів пристроїв:

драйвери апаратних пристроїв, які управляють (через HAL) обладнанням, записують на них дані, що виводяться, і отримують дані, що вводяться, від фізичного пристрою або з мережі. Є безліч типів таких драйверів – драйвери інтерфейсів, пристроїв масової пам'яті тощо;

драйвери файлової системи – це драйвери Windows, що приймають запити на файлове введення-виведення і транслюють їх у запити введення-виведення для конкретного пристрою;

драйвери фільтра файлової системи, які забезпечують дзеркалювання і шифрування дисків, перехоплення введення-виведення і деяке додаткове оброблення інформації перед передаванням її на наступний рівень;

мережеві редиректори та сервери, що є драйверами файлових систем, які передають запити файлової системи на введення-виведення іншим комп'ютерам у мережі та приймають від них аналогічні запити;

драйвери протоколів, що реалізовують мережеві протоколи, наприклад, TCP/IP, NETBEUI і IPX/SPX;

драйвери потокових фільтрів ядра, що діють по ланцюжку для оброблення потокових даних, наприклад під час запису і відтворення аудіо- і відеоінформації.

Оскільки встановлення драйвера пристрою – це єдиний спосіб додавання в систему стороннього коду режиму ядра, деякі програмісти пишуть драйвери просто для того, щоб дістати доступ до внутрішніх функцій або структур даних операційної системи, недоступних із призначеного для користувача режиму (але документованих і підтримуваних у DDK).

Зараз усі драйвери до всіх пристроїв для Windows написані на технології INF. 2000 року Microsoft зробила великий крок у питанні встановлення додатків і "вигадала" MSI – архів cabinet-формату, до якого пришиті настановні скрипти в бінарному форматі з GUI-діалогами.

Скрипт – це текстовий файл, що містить секції, параметри секцій і значення параметрів секцій, що описують дії, які необхідно виконати інтерпретаторові скрипту.

У системі має бути встановлено **MSI Installer**, що є сервісом із правами системи і що тим самим дозволяє встановлювати багато пакетів навіть, із правами користувача. Основними особливостями MSI-інсталера, успадкованими від технології Active Setup з INF, є можливість створювати точки відкоту реєстру, які застосовуються під час деінсталяції пакета автоматично. Це було покликано у спробі позбавитися від проблеми постійного руйнування і засмічення системного реєстру – великої проблеми у Windows. Відмітною особливістю MSI є властивість створювати в реєстрі десятки унікальних **GUID** під кожний додаток і навантажувати їх купою параметрів. *Скрипти INF нічим не відрізняються від простих bat-файлів* за принципом, якщо задано послідовність дій, інтерпретатор виконує або виводить помилку. Скрипти уміють працювати з реєстром, файлами, папками, INI-, LNK-файлами, виводити простенькі діалоги, і послідовно запускати PE-файли на виконання. У принципі для встановлення додатків цього вистачає.

У Windows наявні за замовчуванням два інтерпретатори скриптів INF: **SETUPAPI** і **ADVANCEDINF**. *Інтерпретатор* – це програма, що розпізнає синтаксис скрипту і послідовно виконує команди, указані у скрипті.

Обидва інтерпретатори становлять два DLL-файли в системній директорії та деяка кількість ключів у реєстрі. Інтерпретатор SETUPAPI перебуває в бібліотечному файлі **setupapi.dll**, інтерпретатор ADVANCEDINF – у бібліотечному файлі **advpack.dll**. Виходячи з того, що бібліотеки інтерпретаторів

не є виконуваними файлами, потрібний зовнішній ініціатор запуску функції інтерпретації скрипту. Ним є системна утиліта **RunDLL32.exe**.

Формат запуску будь-якої бібліотеки за допомогою RunDLL32:

```
rundll32.exe libraryname,EntryPoint parameters,
```

де *libraryname* – ім'я файла бібліотеки (у цьому разі бібліотеки інтерпретатора), допустимо вказувати без розширення, якщо бібліотеку зареєстровано у списку SharedDLLs у системному реєстрі;

EntryPoint – реєстрочутливе ім'я точки входу в бібліотеку (ім'я функції, що викликається), указано відразу після коми, без пропусків;

parameters – параметри, передавані функції.

Синтаксис командного рядка для запуску інтерпретаторів скриптів INF побудовано на цих же принципах. Формат запуску обумовлено високим ступенем дозволеності дій у системі файла простого текстового формату. Microsoft не звикла довіряти скриптам. Тому простий запуск подвійним клацанням у системі за замовчуванням відкриває файл скрипту блокнотом. Це достатньо легко змінити.

SETUPAPI – це основний інтерпретатор. Виконує основну безліч дій:

запис і видалення ключів, параметрів і значень системного реєстру;

додавання і видалення рядків, заміна значень INI-файлів;

розпаковування файлів із CAB-архівів, копіювання та видалення файлів;

перейменування, зміна атрибутів файлів і папок;

установлення драйверів;

створення системних сервісів і пристроїв (Windows NT-based);

перевірка призначених для користувача повноважень.

Типовий приклад запуску інтерпретатора SETUPAPI для виконання скрипту:

```
rundll32.exe setupapi,InstallHinfSection DefaultInstall
                    132 C:\Script.inf,
```

де *InstallHinfSection* – ім'я функції, що викликається (точка входу);

DefaultInstall – перший параметр для функції, що викликається, означає ім'я виконуваної секції в INF-скрипті;

132 – другий параметр для функції, що викликається, прапорець для оброблення скрипту;

C:\Script.inf – третій параметр для функції, що викликається, повний шлях до файла скрипту.

Зверніть увагу, потрібний саме повний шлях, оскільки проста вказівка імені файла має на увазі розташування файла скрипту в системній директорії Windows. Ця ж примітка однаковою мірою належить і до інтерпретатора ADVANCEDINF.

ADVANCEDINF – це надбудовний над SETUPAPI інтерпретатор, що дозволяє виконувати додаткові функції. Стандартні функції передає на виконання інтерпретатору SETUPAPI. Функції, підтримувані інтерпретатором ADVANCEDINF, окрім зазначених:

попередній запис змінних ключів реєстру в бінарний файл (функція відкоту);

одноразове виконання дій під кожним користувачем (довстановлення) під час інсталяції та деінсталяції під час входу в систему (Active Setup);

запуск виконуваних файлів із параметрами у прихованому та нормальному режимах;

виведення простих діалогових вікон;

читання директорій призначення операцій із файлами з реєстру.

Виходячи зі згаданого раніше, буде слушним віддавати перевагу інтерпретатору ADVANCEDINF: так можна користуватися перевагами обох інтерпретаторів. Виняток становить установлення драйверів, тут із цим може справлятися тільки SETUPAPI.

Типовий приклад запуску інтерпретатора ADVANCEDINF для виконання скрипту:

```
rundll32.exe advpack,LaunchINFSection
   C:\Script.inf,DefaultInstall,4,
```

де *LaunchINFSection* – точка входу;

C:\Script.inf – перший параметр для функції, що викликається, повний шлях до файла скрипту;

DefaultInstall – другий параметр, ім'я виконуваної секції в INF-скрипті (**зверніть увагу**, ім'я секції нечутливе до регістра, на відміну від точки входу);

4 – прапорець реакції інтерпретатора під час оброблення команд скрипту (табл. 10.1).

Таблиця 10.1

Значення прапорця реакції інтерпретатора під час оброблення команд скрипту

Числові значення	Опис
0 або 128	Не перезавантажувати комп'ютер
1 або 129	Обов'язково і без питань перезавантажити комп'ютер
2 або 130	Запитати в користувача: перезавантажити комп'ютер чи ні
3 або 131	Якщо потрібно перезавантажувати комп'ютер, перезавантажити без питань
4 або 132	Якщо потрібно перезавантажувати комп'ютер, запитати в користувача: перезавантажувати чи ні

Для розпізнавання текстового файла як файла з INF-структурою інтерпретатор шукає в непорожніх рядках секційний заголовок [Version]. Заголовок традиційно розташовано на початку файла, і він містить у своїй секції декілька рядків, що визначають тип скрипту. Заголовки бувають таких типів:

1. Стандартні.
2. Драйверні.
3. Розширені.

Тип скрипту вказує, яким саме інтерпретатором слід виконувати скрипт. Залежно від інтерпретатора, можна виконувати різний набір дій.

Стандартний заголовок. Скрипт зі стандартним заголовком призначено для виконання операцій загального призначення. Набір функцій, підтримуваних скриптом зі стандартним заголовком, визначається інтерпретатором, якому було передано на виконання цей скрипт.

Ось секція стандартного заголовка скрипту:

```
[Version]
Signature="$CHICAGO$"
SetupClass=BASE
ClassGUID={00000000-0000-0000-0000-000000000000}
```

Указані тут параметри SetupClass і ClassGUID рідко використовують разом, звичайно достатньо будь-якого з них. Із драйверними заголовками ситуація дещо інша. Обмеження для використання стандартного заголовка: украй не рекомендовано використовувати під час написання

скрипту встановлення драйверів. Під час побудови списку драйверів, відповідних типу встановлюваного пристрою, інтерпретатор відбирає скрипти драйверів саме за заголовком і драйверні скрипти зі стандартним заголовком ніколи не буде уміщено до списку.

Драйверний заголовок уміє обробляти тільки інтерпретатор SETUPAPI.

Приклад драйверного заголовка:

```
[Version]
Signature="$CHICAGO$"
Class=System
ClassGuid={4d36e97d-e325-11ce-bfc1-08002be10318}
CatalogFile=sample.cat
Provider=%MSFT%
LayoutFile=layout.inf
DriverVer=03/16/2019,6.00.9830.1
```

Визначенням драйверного заголовка є параметри Class і ClassGuid. Вони вказують на один і той самий тип драйвера, *Class* показує ім'я типу, *ClassGuid* – його GIUD. Список відомих поточній операційній системі типів драйверів (табл. 10.2) можна так знайти в системному реєстрі:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Class.

Таблиця 10.2

Класи драйверних типів

GUID	Класи
{4D36E966-E325-11CE-BFC1-08002BE10318}	Computer
{4D36E968-E325-11CE-BFC1-08002BE10318}	Display
{4D36E96B-E325-11CE-BFC1-08002BE10318}	Keyboard
{4D36E96A-E325-11CE-BFC1-08002BE10318}	HDC
{745A17A0-74D3-11D0-B6FE-00A0C90F57DA}	HID
{6BDD1FC6-810F-11D0-BEC7-08002BE2092F}	Image
{4D36E96C-E325-11CE-BFC1-08002BE10318}	Media
{4D36E96D-E325-11CE-BFC1-08002BE10318}	Modem
{4D36E96E-E325-11CE-BFC1-08002BE10318}	Monitor
{4D36E96F-E325-11CE-BFC1-08002BE10318}	Mouse
{50906CB8-BA12-11D1-BF5D-0000F805F530}	MultiPortSerial
{4D36E972-E325-11CE-BFC1-08002BE10318}	Net
{4D36E979-E325-11CE-BFC1-08002BE10318}	Printer
{4D36E97B-E325-11CE-BFC1-08002BE10318}	SCSI Adapter
{50DD5230-BA8A-11D1-BF5D-0000F805F530}	Smart Cart Reader

Не рекомендовано вказувати тільки Class, оскільки не всі системи цим задовольняться. Під час визначення скрипту як драйверного, у системі ініціюється "вакцина" – реакція на зміну системної частини реєстру. Водночас, залежно від поточних налаштувань, може бути виведено запит на підтвердження встановлення драйвера. Починаючи із Microsoft Windows XP (NT 5.1), під час налаштування за замовчуванням створюється точка відновлення системної частини реєстру і змінених у ході встановлення системних файлів. Крім того, у разі вказування драйверного заголовка скрипт буде позбавлено можливостей, підтримуваних в AdvancedINF.

Розширений заголовок скрипту вказує, яка версія інтерпретатора AdvancedINF необхідна для його виконання. Якщо поточна версія інтерпретатора нижча від вказаного у скрипті, оброблення скрипту буде перервано і виведено помилку – Error message:

```
[Version]
Signature="$CHICAGO$"
AdvancedINF=2.0, "Error message"
```

Якщо файл із розширеним заголовком, призначений для оброблення інтерпретатором AdvancedINF, буде в командному рядку передано на виконання для SETUPAPI, виконано його буде як файл зі стандартним заголовком.

У кожному типі заголовка незмінним рядком є параметр Signature. Його значення пояснює, для якої операційної системи призначено скрипт. Відомо два значення: **$CHICAGO$** і **$WINDOWS NT$**.

Перший призначено для всіх операційних систем Windows; другий тільки для Windows NT 5.0 (2000), 5.1 (XP), 5.2 (2003). Виняток становлять драйверні скрипти, тут необхідно по можливості більш точно визначити тип системи, інакше драйвер (залежно від типу) не буде розпізнано як відповідний.

Драйвер містить знання того, які файли він використовує. Часто визначити, чи потрібний драйвер (сервіс) можна, просто переглянувши, які файли він використовує. Одним із якнайкращих вирішень цього питання є програма **Dependency Walker** (переглядання залежностей) – depends.exe. Dependency Walker – це дуже корисна програма, що дозволяє дізнатися, які DLL потрібні для запуску якого-небудь виконуваного файла (і побачити все дерево залежностей), а також дізнатися, які функції він із них імпортує. Крім того, можна подивитися таблиці імпорту й експорту у DLL.

Головне вікно показано на рис.10.2. Так, наприклад, наведено інформацію про структуру бібліотеки kernel32.dll ОС Windows.

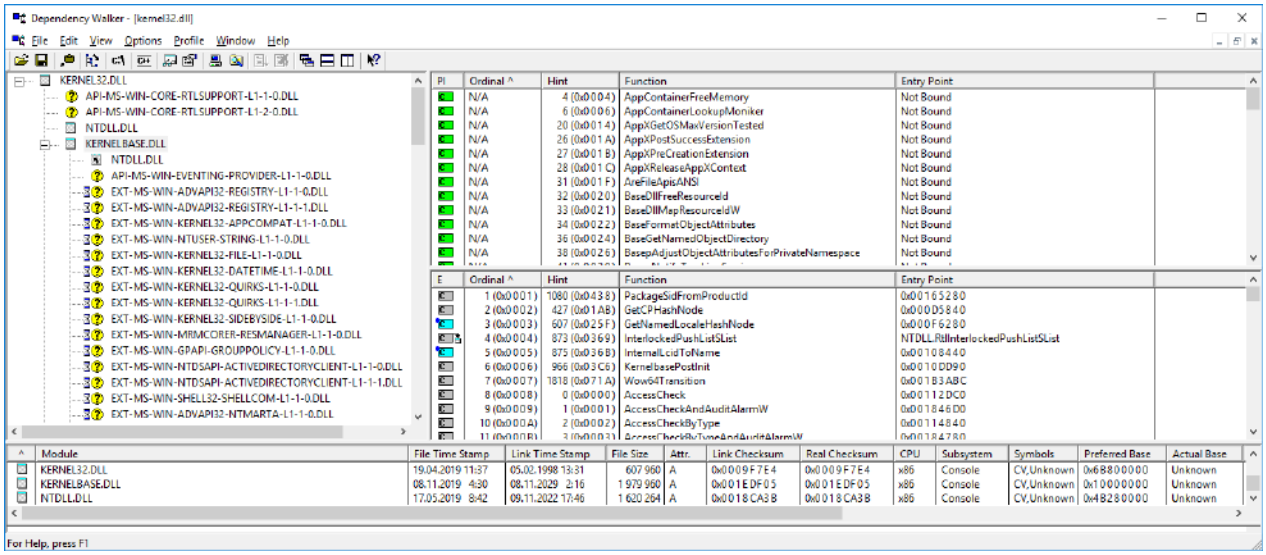


Рис. 10.2. **Основне вікно програми Dependency Walker**

Щоб побачити, які бібліотеки DLL, такі як елементи управління ActiveX, перебувають у процесі динамічного завантаження драйвера, скористайтеся **функцією профілізації** depends.exe. Потім перевірте додаток, щоб переконатися, що було задіяно шляхи всіх кодів. Після завершення профілізації depends.exe відображає бібліотеки DLL, які були динамічно завантажені.

Використовуючи Dependency Walker потрібно пам'ятати, що бібліотека DLL може залежати від іншої бібліотеки DLL або від її конкретної версії. Depends.exe можна використовувати на комп'ютері розробника або на кінцевому комп'ютері. На комп'ютері розробника depends.exe відображає бібліотеки DLL, потрібні для підтримки додатку. Якщо під час запуску додатка на кінцевому комп'ютері виникають проблеми, можна скопіювати Dependency Walker на кінцевий комп'ютер і відкрити додаток у depends.exe. Dependency Walker відображає бракувальні бібліотеки DLL, а також бібліотеки несумісних версій.

Після визначення повного списку бібліотек DLL, від яких залежить певний додаток, можна зрозуміти, які із цих бібліотек необхідно поширювати разом із додатком під час його розгортання на іншому комп'ютері. Здебільшого немає необхідності в поширенні системних бібліотек DLL.

Загальне завдання для виконання

1. Аналітична частина

Зробіть аналіз структури одного з inf-файлів, установленого в операційній системі в директорії %SYSTEMROOT%:\ WINDOWS\inf, і заданого в п. 1 індивідуального завдання. Аналіз має містити опис секцій цього inf-файла. Водночас слід урахувати, що в разі відсутності вказаного в п. 1 файла у вашій операційній системі слід вибрати його аналог.

2. Дослідна частина

Етап 1. Вивчіть властивості встановленого драйвера в операційній системі в директорії %SYSTEMROOT%:\WINDOWS\system32\drivers і заданого в п. 2 індивідуального завдання.

До властивостей, відображених у звіті слід зарахувати такі параметри:

внутрішнє ім'я драйвера;

ім'я драйвера, що відображається;

тип драйвера;

тип запуску драйвера;

контроль за помилками;

група драйверів (сервісів), до якої належить заданий драйвер;

тег;

шлях до файла драйвера;

розмір файла драйвера;

обліковий запис, від імені якого запускають драйвер;

порядок відновлення системи після перебою драйвера.

Етап 2. Побудуйте дерево залежностей указаного в п. 2 драйвера, від системних бібліотек, використовуючи утиліту depends.exe.

3. Програмна частина – додаткове завдання (табл. 10.3)

Використовуючи рекомендовану літературу, подайте в консольному або графічному вигляді заданий в п. 3 індивідуального варіанта додаток. Передбачте відомості про автора, групу і факультет.

Таблиця 10.3

Індивідуальні завдання для виконання

Варіанти	Inf-файл	Драйвери	Додаткове завдання
1	2	3	4
1	multiprt.inf	fltMgr.sys	Створіть додаток із графічним інтерфейсом. Програми має запускати диспетчер для власного сервісу. Перед тілом функції оголошено всі глобальні імена, які надалі використовують у сервісі. Водночас сервіс має відстежувати: реєстрацію обробника керівних команд для сервісу; ініціалізацію структури стану сервісу
2	net1394.inf	nwlnkflt.sys	Створіть додаток, що встановлює зв'язок із диспетчером сервісів і дозволяє відкривати доступ до бази даних сервісів. Тип доступу до бази даних має передбачати будь-яку комбінацію прапорців, зокрема й установлення родових прав доступу
3	memstpci.inf	asyncmac.sys	Розробіть графічний інтерфейс, що дозволяє виводити в діалоговому вікні властивості довільно вибраної користувачем служби зі списку зареєстрованих на цьому локальному комп'ютері. Водночас обов'язково слід забезпечити такі параметри: тип облікового запису, від імені якого запущено службу; дії, що виконують зі службою під час виникнення першого, другого і подальшого перебою. За потреби врахуйте та надайте опис вибраного сервісу на комп'ютері
4	hidbth.inf	ipfltdrv.sys	Створіть додаток із графічним інтерфейсом. Програми має запускати диспетчер для власного сервісу. Перед тілом функції оголошено всі глобальні імена, які надалі використовують у сервісі. Водночас сервіс має відстежувати відкриття файла протоколу роботи сервісу
5	mfcem33.inf	tcpip6.sys	Створіть додаток із графічним інтерфейсом. Програми має запускати диспетчер для власного сервісу. Перед тілом функції оголошено всі глобальні імена, які надалі використовують у сервісі. Водночас сервіс має відстежувати ініціалізацію структури стану сервісу

Продовження табл. 10.3

1	2	3	4
6	rspndr.inf	wudfpf.sys	Створіть додаток із графічним інтерфейсом, який дозволяє управляти правами доступу до довільно вибраного із зареєстрованих у системі системних служб. Забезпечте підтримку всіляких прапорців service_, read_, write_. Реалізуйте можливість зміни прапорців у процесі роботи програм
7	dot4prt.inf	raspppoe.sys	Розробіть графічний додаток, що дозволяє встановлювати призначені для користувача сервіси. Усередині сервісу слід забезпечити довільні імена, шляхи, окремий процес, особисту групу. Ці параметри мають встановлювати в **налаштуваннях** програми. Забезпечте можливість видалення **тільки** встановлених призначених для користувача сервісів
8	dfrg.inf	mrxdav.sys	Розробіть консольний додаток, що дозволяє управляти заздалегідь установленою призначеною для користувача службою. Під управлінням розуміють старт, стоп, паузу у виконанні, відновлення, рестарт цієї служби. Супроводжуйте вказані дії системою повідомлень і підказувань
9	digiasyn.inf	audstub.sys	Розробіть графічний додаток, у якому продемонструйте визначення внутрішнього імені сервісу за його зовнішнім ім'ям за допомогою функції GetServiceKeyName. Під внутрішнім ім'ям сервісу розуміють те ім'я, під яким сервіс зберігають у базі даних сервісів, а також використовується диспетчером сервісів для посилань на цей сервіс
10	netepvcp.inf	ndistapi.sys	Розробіть консольний додаток визначення стану сервісу за допомогою виклику функції QueryServiceStatus. Для цього необхідно зв'язатися із Service Control Manager із підключенням активної бази даних сервісів, відкриття сервісу, заздалегідь установленого в системі, визначення стану сервісу і розшифрування цього стану шляхом аналізу структури _SERVICE_STATUS
11	ramdisk.inf	netbt.sys	Розробіть графічний додаток визначення стану сервісу за допомогою виклику функції QueryServiceStatus. Для цього необхідно зв'язатися із Service Control Manager із підключенням активної бази даних сервісів, відкриття сервісу, заздалегідь установленого в системі, визначення стану сервісу і розшифрування цього стану шляхом аналізу структури _SERVICE_STATUS

Продовження табл. 10.3

1	2	3	4
12	ppa.inf	aec.sys	Створіть додаток із консольним інтерфейсом, що дозволяє визначати конфігурацію сервера. Під конфігурацією сервісу розуміють інформацію, яка описує внутрішні й зовнішні імена сервісу, режими його запуску і роботи, використовуючи функцію QueryServiceConfig. За наслідками аналізу конфігурації виведуть таку інформацію: тип сервісу, тип запуску, управління помилками сервісу, ім'я шляху; ім'я, що відображається
13	oem0.inf	ip6fw.sys	Розробіть графічний додаток, що дозволяє управляти заздалегідь установленою призначеною для користувача службою. Під управлінням розуміють старт, стоп, паузу у виконанні, відновлення, рестарт цієї служби. Супроводжуйте вказані дії виведенням відповідних повідомлень у робочій зоні вікна
14	bthprint.inf	ipinip.sys	Розробіть графічний інтерфейс, що дозволяє виводити в діалоговому вікні властивості довільно вибраної служби користувача зі списку зареєстрованих на цьому локальному комп'ютері. Водночас обов'язково слід виводити такі параметри: внутрішнє ім'я служби; параметри, що відображаються; тип сервісу, тип запуску, прапорці контролю за помилками; групу, до якої належить сервіс; шлях до файла сервісу, а також його опис
15	bthspp.inf	psched.sys	Розробіть графічний додаток, що дозволяє блокувати та розблоковувати процесу базу даних сервісів, а також визначати: заблоковано базу даних сервісів чи ні. Продемонструйте факт блокування бази шляхом створення нового процесу, який має здійснювати спробу доступу. Ця спроба має завершитися успішно в розблокованому стані й неуспішно – у протилежному випадку
16	swflash.inf	nwlnkfwd.sys	Створіть додаток із графічним інтерфейсом, що дозволяє змінювати конфігурацію сервера. Забезпечте управління режимами роботи програми за допомогою команд меню. Під конфігурацією сервісу розуміють інформацію, яка описує внутрішні й зовнішні імена сервісу, режими його запуску і роботи, використовуючи функції ChangeServiceConfig. За наслідками аналізу конфігурації виведіть таку інформацію: тип сервісу, тип запуску, управління помилками сервісу, ім'я шляху; ім'я, що відображається

Продовження табл. 10.3

1	2	3	4
17	msrio.inf	ptilink.sys	Створіть додаток, що управляє прапорцями SC_ типів доступу до Service Control Manager. Передбачте всі можливі комбінації прапорців. Реалізуйте приклад взаємодії керівні програми із Service Control Manager
18	msnike.inf	ndiswan.sys	Створіть додаток із графічним інтерфейсом, який дозволяє управляти правами доступу до довільно вибраного із зареєстрованих у системі системних служб (драйверів). Забезпечте підтримку таких прав доступу, як service_change_config, service_enumerate_dependents service_interrogate, service_pause_continue, service_query_config, service_query_status, service_start, service_stop, service_user_defined_control
19	mdmrisa.inf	rasl2tp.sys	Створіть додаток із графічним інтерфейсом. Програми має запускати диспетчер для власного сервісу. Перед тілом функції оголошено всі глобальні імена, які надалі використовують у сервісі. Водночас сервіс має відстежувати встановлення стану сервісу
20	netlanep.inf	atmarpc.sys	Розробіть графічний додаток, у якому продемонструйте визначення зовнішнього імені сервісу за його внутрішнім ім'ям за допомогою функції GetServiceDisplayName. Під зовнішнім ім'ям сервісу розуміють те ім'я, яке використовують для посилань на сервіс у вікні управління сервісами
21	ntapm.inf	raspti.sys	Розробіть графічний інтерфейс, що дозволяє виводити в діалоговому вікні властивості довільно вибраного драйвера зі списку зареєстрованих на цьому локальному комп'ютері. Водночас обов'язково слід виводити такі параметри: внутрішнє ім'я драйвера; параметри, що відображаються; тип драйвера, тип запуску, прапорці контролю за помилками; групу, до якої належить драйвер, а також його опис
22	ovcomp.inf	mrxsmb.sys	Створіть додаток із графічним інтерфейсом. Програми має запускати диспетчер для власного сервісу. Перед тілом функції оголошено всі глобальні імена, які надалі використовують у сервісі. Водночас сервіс має відстежувати: реєстрацію обробника команд, що управляють, для сервісу; ініціалізацію структуру стану сервісу

Продовження табл. 10.3

1	2	3	4
23	srchasst.inf	ipnat.sys	Створіть додаток, що встановлює зв'язок із диспетчером сервісів і дозволяє відкривати доступ до бази даних сервісів. Тип доступу до бази даних має передбачати будь-яку комбінацію прапорців, зокрема установлення родових прав доступу
24	netauni.inf	afd.sys	Розробіть графічний інтерфейс, що дозволяє виводити в діалоговому вікні властивості довільно вибраної користувачем служби зі списку зареєстрованих на цьому локальному комп'ютері. Водночас обов'язково слід забезпечити такі параметри: тип облікового запису, від імені якої запущено службу; дії, що виконують зі службою під час виникнення першого, другого і подальшого перебою. Дайте можливість користувачеві самостійно встановлювати час (у с), через який буде відбуватися обнулення лічильника відмов, а також рестарт сервісу і рестарт комп'ютера
25	mscpqpa1.inf	secdrv.sys	Створіть додаток із графічним інтерфейсом, який дозволяє управляти правами доступу до довільно вибраного із зареєстрованих у системі системних служб. Забезпечити підтримку всіляких прапорців service_, read_, write_. Реалізуйте можливість зміни прапорців у процесі роботи програми
26	usbprint.inf	rspndr.sys	Розробіть графічний додаток, що дозволяє встановлювати призначені для користувача сервіси. Усередині сервісу слід забезпечити довільні імена, шляхи, окремий процес, особисту групу. Ці параметри мають встановлювати в **налаштуваннях** програми. Забезпечте можливість видалення **тільки** встановлених призначених для користувача сервісів
27	adm_port.inf	rasacd.sys	Створіть додаток із графічним інтерфейсом. Програми має запускати диспетчер для власного сервісу. Перед тілом функції оголошено всі глобальні імена, які надалі використовують у сервісі. Водночас сервіс має відстежувати відкриття файла протоколу роботи сервісу

Закінчення табл. 10.3

1	2	3	4
28	epstw2k.inf	wudfrd.sys	Розробіть консольний додаток, що дозволяє управляти заздалегідь установленою призначеною для користувача службою. Під управлінням розуміють старт, стоп, паузу у виконанні, відновлення, рестарт цієї служби. Супроводжуйте вказані дії системою повідомлень і підказувань
29	vgx.inf	ndisuio.sys	Розробіть консольний додаток визначення стану сервісу за допомогою виклику функції QueryServiceStatus. Для цього необхідно зв'язатися із Service Control Manager із підключенням активної бази даних сервісів, відкриття сервісу, заздалегідь установленого в системі, визначення стану сервісу і розшифрування цього стану шляхом аналізу структури _SERVICE_STATUS
30	netgpc.inf	netbios.sys	Створіть додаток із графічним інтерфейсом. Програми має запускати диспетчер для власного сервісу. Перед тілом функції оголошено всі глобальні імена, які надалі використовують у сервісі. Водночас сервіс має відстежувати ініціалізацію структури стану сервісу

Контрольні запитання

1. Що таке "служба"?
2. Назвіть основні операції під час роботи зі службою з боку користувача.
3. Назвіть стандартні способи взаємодії зі службою.
4. Поясніть порядок використання GUID.
5. Укажіть гілки системного реєстру для зберігання записів служб.
6. Назвіть системний сервіс, що забезпечує роботу з іншими службами, їхній запуск, зупинку.
7. Дайте характеристику інтерпретатора SETUPAPI.
8. Який системний процес дозволяє виконати команди інтерпретатора SETUPAPI?
9. Поясніть призначення сигнатури секції стандартного заголовка inf-файла.

10. Які функції покладають на Диспетчер системних служб?

11. Які системні динамічні бібліотеки реалізують функції API під час управління службами?

12. До якого класу файлів належать msi-архіви?

13. У чому полягає відмінність сигнатури $CHICAGO$ від $WINDOWS NT$?

14. Що таке Entry point для dll-модулів у програмі Dependency Walker?

15. Назвіть властивість служби, яку задає тег.

Лабораторна робота 11
Дослідження засобів захисту даних

Мета роботи: набуття практичних навичок в організації захисту файлів в операційній системі, здійсненні криптоаналізу, аудиту парольного захисту різних об'єктів операційної системи. Ознайомлення з реалізацією шифрування, генерацією ключів, створення і перевірки цифрових підписів та інших криптографічних завдань засобами інтерфейсу CryptoAPI, а також набуття практичних навичок в організації та зберіганні відкритих, особистих або сесійних ключів.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно вивчити порядок шифрування і дешифрування даних засобами операційної системи, механізми автентифікації та цифрового підпису.

Додаткову інформацію під час підготовки до роботи можна отримати в [11].

Теоретичні відомості

Шифрування – це спосіб перетворення відкритої інформації в закриту й назад. Застосовують для зберігання важливої інформації в ненадійних джерелах або передавання її незахищеними каналами зв'язку. Сенс шифрування полягає в запобіганні перегляданню початкового змісту

повідомлення тими, у кого немає засобів його дешифрування. Зашифрувати можна не тільки текст, але й різні дані від файлів баз даних і текстових процесорів до файлів зображень.

Захисна підсистема комп'ютера потребує реалізації ряду загальних функцій, пов'язаних із логічним перетворенням змісту об'єктів (функції логічного захисту). До таких функцій належать:

алгоритми контролю за цілісністю об'єктів комп'ютерної системи;

алгоритми автентифікації або авторизації суб'єктів (користувачів, які управляють суб'єктами);

алгоритми підтримки конфіденційності змісту об'єктів (наприклад, об'єкта вторинної автентифікації користувачів).

Міжнародні й національні стандарти описують ряд добре вивчених функцій захисного характеру, зокрема:

алгоритми гешування MD2 і MD5;

алгоритми генерації та перевірки електронного цифрового підпису DSS.

Усі ці алгоритми мають різну специфікацію викликів (зокрема, різну довжину аргументів) і, природно, логічно не сумісні між собою.

Під час організації територіально розподілених комп'ютерних систем у різних локальних сегментах можуть використовуватися функціонально однакові, але семантично різні функції логічного захисту (наприклад, обчислення функцій контролю за цілісністю можна виконувати із застосуванням різних алгоритмів). Особливо актуальна ця проблема щодо стандартизованих і сертифікованих апаратних модулів типу FORTEZZA або Crypton.

Завдання використання зовнішніх функцій логічного захисту актуальне не тільки для захисних суб'єктів, але й для довільного суб'єкта, що входить до комп'ютерної системи (наприклад, використання суб'єкта обчислення електронного цифрового підпису для фіксації цілісності інформації, передаваної в зовнішню мережу). У зв'язку із цим, питання взаємодії з функціями логічного захисту будуть вивчати без прив'язування до конкретних функцій суб'єкта.

Суб'єкти (програми, процеси) комп'ютерної системи, пов'язані з виконанням захисних функцій, можуть використовувати деяку загальну підмножину функцій логічного перетворення об'єктів (зокрема, алгоритми шифрування і контролю за цілісністю об'єктів).

Під час проектування і реалізації суб'єктів комп'ютерної системи підхід, що історично склався, до використання загального ресурсу пов'язано

із використанням суб'єктів, що розподіляють, виконують загальні для деякої підмножини зовнішніх щодо цього суб'єктів функцій (наприклад, динамічно завантажуваних бібліотек – типу DLL для ОС Windows). Логічно поширити цей підхід на функції реалізації захисту.

Проблему проектування суб'єктів, що реалізовують функції логічного захисту, можна розглядати в кількох аспектах:

1. Проблему їхньої *оптимальної реалізації* в межах деякого суб'єкта комп'ютерної системи, до якого звертається решта суб'єктів за виконанням відповідних функцій (у цьому разі мова йде про задавання оптимізації параметрів "швидкодія – пам'ять" під час реалізації захисних функцій).

2. Проблему *мобільності* суб'єктів, що використовують захисні функції під час зміни внутрішнього наповнення, що реалізовує захисні функції суб'єкта (мають на увазі задавання максимальної стерпності суб'єктів, що використовують захисні функції, на інші алгоритми, наприклад інший алгоритм шифрування).

3. Проблему *коректного використання* суб'єкта, що реалізовує захисні функції, із боку зухвалих його модулів (проблема коректного використання дещо ширша, ніж просто коректність суб'єктів, оскільки передавання інформації до асоційованих об'єктів-даних під час виклику функцій логічного захисту (або під час повернення управління) має на увазі вплив на асоційовані об'єкти зухвалого суб'єкта).

У тексті використовують термін **"блоб"** – це блок послідовних даних, пов'язаний з експортом (виведенням у зовнішнє середовище) та імпортом (уведенням із зовнішнього середовища) ключів. У цьому разі модуль реалізації захисних функцій, згідно з термінологією Microsoft, називають *криптопровайдером* – Cryptographic Service Provider (CSP).

Є різні методи шифрування, що використовують спеціальні алгоритми.

З основних методів шифрування можна виділити симетричне й асиметричне. В обох випадках, знаючи алгоритм шифрування, потрібно мати ключ (частину коду), а в останньому випадку, потрібно мати два ключі (відкритий і закритий).

У цій лабораторній роботі наведено безкоштовні програми для шифрування даних, які допоможуть користувачеві захистити свої дані, використовуючи найрізноманітніші за рівнем складності алгоритми. Використовуючи це програмне забезпечення, можна надійно захистити свої секретні дані від сторонніх.

До програм сторонніх виробників, розгляду яких присвячено лабораторну роботу, належать:

1. Заборонений файл.
2. VSEncryptor.
3. Crypt4Free.
4. Universal Shield.
5. CryptoLab.
6. MEO Encryption Software.
7. AxCrypt.
8. Encrypt Files.

Порівняльний аналіз алгоритмів шифрування, які підтримуються цими програмами, наведено в табл. 11.1.

Таблиця 11.1

Порівняльний аналіз алгоритмів шифрування

Назва алгоритму шифрування	Заборонений файл	VSEncryptor	Crypt4Free	Universal Shield	CryptoLab	MEO Encryption Software	AxCrypt	Encrypt Files
1	2	3	4	5	6	7	8	9
0AES-128		√			√		√	
AES-192		√						
AES-256		√		√				√
RC2		√						√
RC4		√						√
RC5		√						
RC6		√						√
DES		√			√			
3-DES						√		
DESX			√					
Triple DES (168)		√		√				
Triple DES (192)								√
Blowfish		√	√	√	√	√		√
Twofish (128)		√						
Twofish (256)								√

Закінчення табл. 11.1

1	2	3	4	5	6	7	8	9
Serpent (128)		√						
Serpent (256)				√				√
Affine Shift					√			
ARCFOUR					√			
Caesar					√			
Camelia		√						
Cast (128)				√				
Cast (256)		√						√
Cobra128				√				
Enigma					√			
GOST		√						
IDEA	√	√						
Ice								√
MARS		√						√
Misty 1								√
Modular Shift Alg.					√			
Rail Fence Algoritm					√			
PC1				√				
Scytale					√			
SEED		√						
SHA-1							√	
SHAKAL-2		√						
Substitution					√			
Skipjack		√						
XTEA		√						√
Vigenure					√			

Заборонений файл – це програма для шифрування файлів. За допомогою неї можна шифрувати окремі файли за алгоритмом IDEA (International Data Encryption Algorithm). Цей алгоритм досить стійкий і надійний, його широко використовують у Європі. Програма може додати свій пункт у меню провідника, щоб можна було більш зручно запускати шифрування (рис. 11.1).

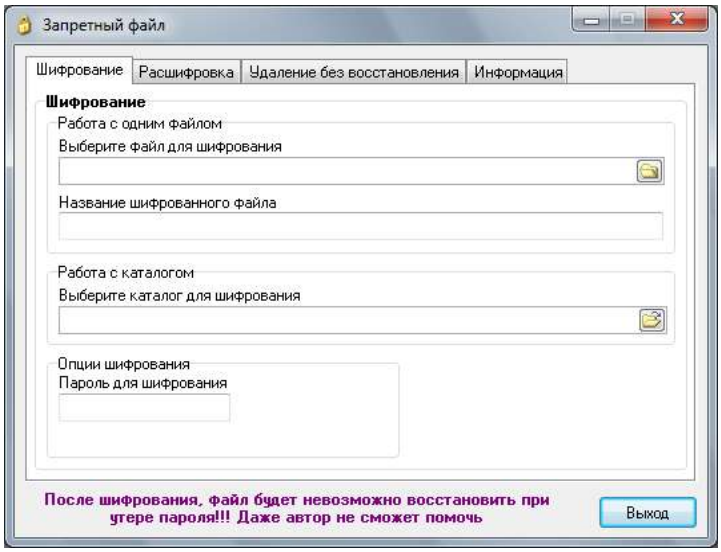


Рис. 11.1. **Головне вікно програми "Заборонений файл"**

VSEncryptor – це програма, призначена для шифрування файла або тексту. Можна зашифрувати будь-який файл або текст за вибором (рис. 11.2).

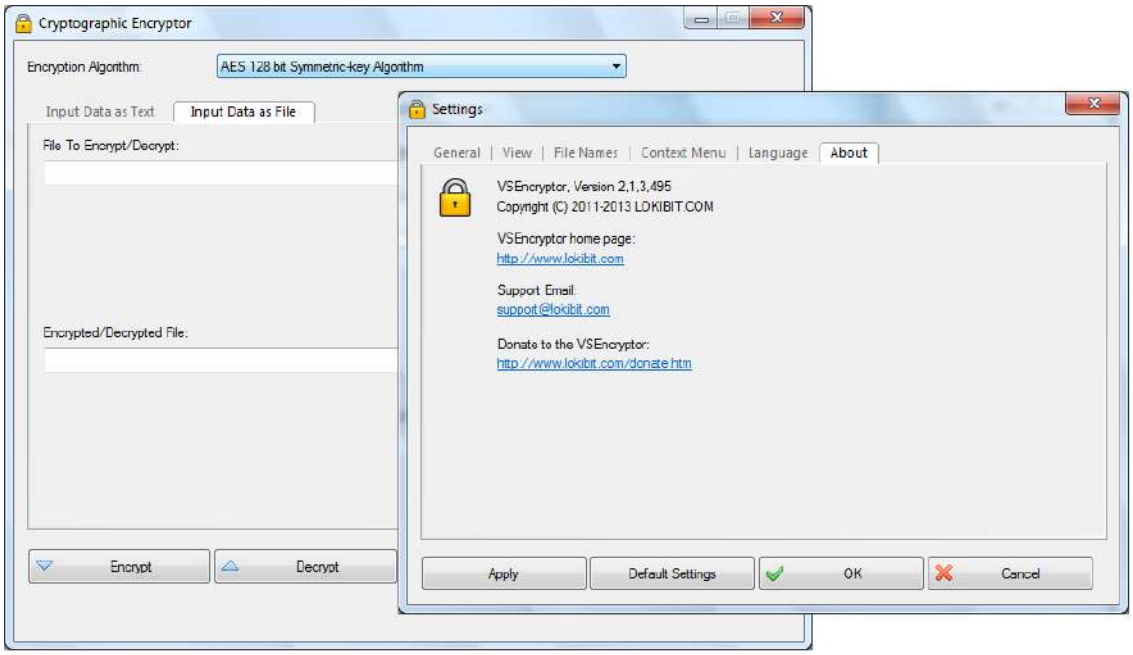


Рис. 11.2. **Головне вікно програми VSEncryptor**

VSEncryptor є серйозним рішенням, але водночас вона є простою у використанні й не потребує яких-небудь технічних знань: просто виберіть (наприклад, перетягнувши файл у належне поле редактора) будь-який файл (або текст) і дайте команду "Шифрувати".

Crypt4Free – це програма, яка дозволяє зашифровувати будь-які файли на будь-якому носієві, використовуючи один із двох наявних у програмі алгоритмів – Blowfish (448-бітовий ключ) і DESX (128-бітовий ключ) (рис. 11.3).

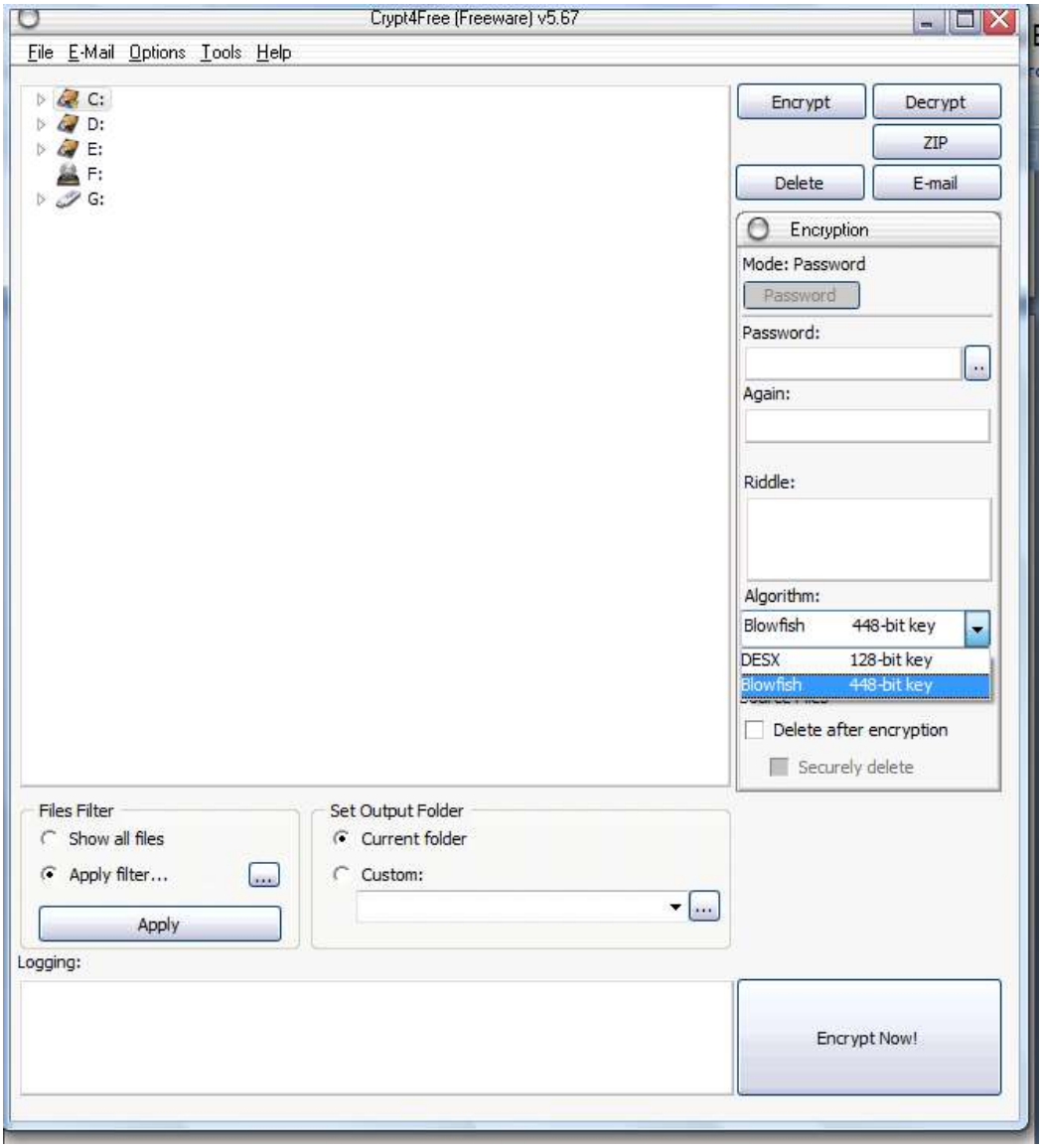


Рис. 11.3. **Головне вікно програми Crypt4Free**

Перед тим як зашифрувати файл, потрібно буде встановити пароль, причому його можна згенерувати автоматично або набрати самому на віртуальній клавіатурі (робиться це для захисту від кейлогерів), після чого натиснути кнопку Start. Підтримується одночасно із шифруванням упакування файлів у zip-архіви; можливе швидке відправлення таких архівів e-mail. Окрім цього, можлива спільна робота з поштовою програмою для відправлення текстових повідомлень у зашифрованому вигляді.

Universal Shield – це утиліта для захисту і шифрування файлів. Дозволяє зашифрувати, приховати файли, папки, диски та встановити додаткові права на доступ.

Universal Shield пропонує 9 алгоритмів шифрування, зокрема Blowfish, TRIPLE-DES, Rijndael тощо. Додаткові функції мають можливість налаштування довірених процесів (програми, які завжди зможуть дістати доступ до захищених файлів), приховування файлів за шаблоном (наприклад *.mpeg) і декілька можливостей обмежувати доступ до системних папок (рис. 11.4).

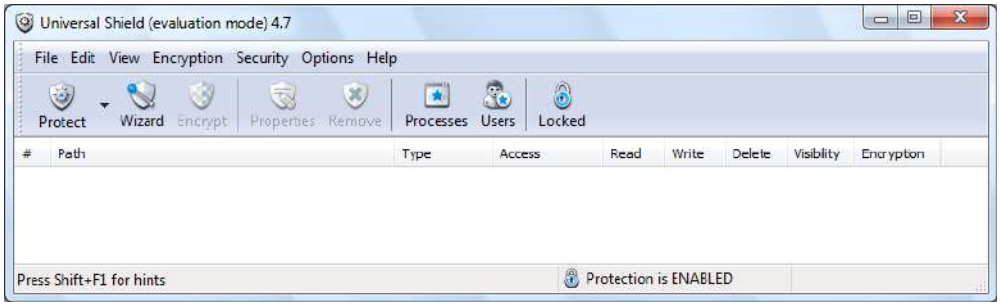


Рис. 11.4. **Головне вікно програми Universal Shield**

CryptoLab – це безкоштовна криптографічна сервісна програма для Microsoft Windows, яка взаємодіє з кодуванням, заснованим на текстах / декодуванням із використанням плагінів шифрових модулів. До покращуваних функцій належать SMTP – підтримка електронною поштою, диспетчер ключів і псевдовипадкова генерація ключів.

До складу CryptoLab введено модулі кодування:

Affine Shift;

ARCFOUR;

Blowfish-шифр;

Caesar-шифр;

Cost;

модулярний алгоритм перемикання (від EJC-криптографії);

Rijndael (американський стандарт кодування).

Substitution-шифр.

Vigenre-шифр.

Із CryptoLab можна кодувати текстові документи з військовими стандартами кодування: такі алгоритми, як Rijndael використовують у кодуванні документів, класифікованих як документи держави США. Використовуючи CryptoLab, можна також установити високий рівень безпеки в комп'ютерній системі (рис. 11.5).

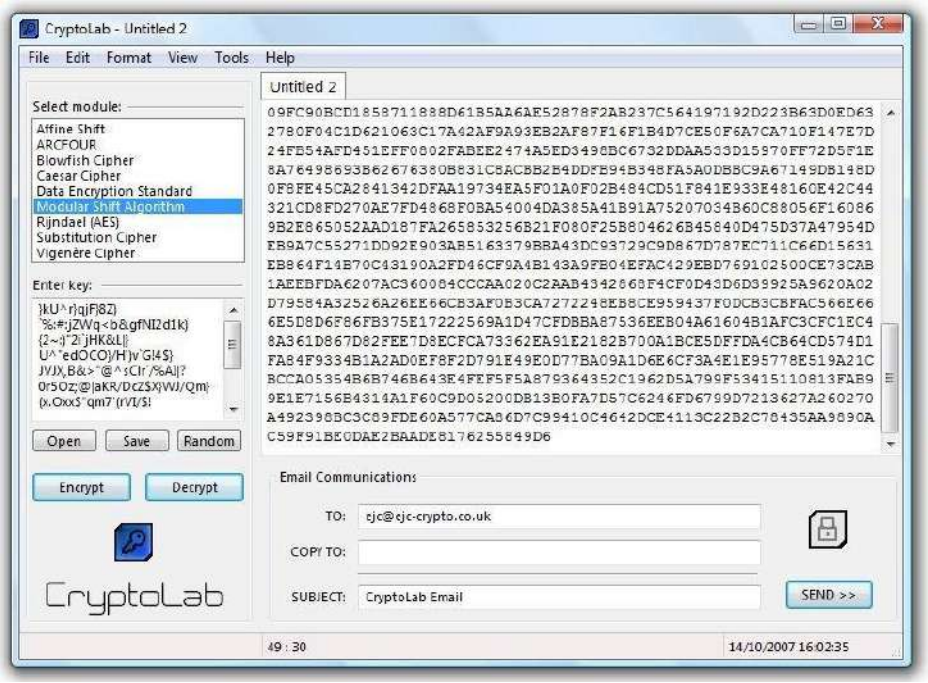


Рис. 11.5. **Головне вікно програми CryptoLab**

Скачування і запуск цієї програми в деяких країнах є нелегальним із-за певних криптографічних законів. EJC Cryptography не може надати легальну пораду.

Примітка. Обов'язково потрібно проглянути відповідні закони країни за цією програмою. Так, наприклад, закони США на момент написання криптографічної програми не може бути експортовано до Афганістану, Куби, Ірану, Іраку, Лівії, Північної Кореї, Судану, Сербії та Сирії.

MEO Encryption Software є програмою для файла, шифрування електронної пошти та дешифрування. У ній є графічний інтерфейс користувача, тільки із трьома опціями. Можна або зашифрувати файли, або зашифрувати електронну пошту, або дешифрувати файл.

Перший дозволяє шифрувати файли для випробування на злом. Можна використовувати два алгоритми шифрування, щоб зашифрувати дані: стандартний шифр або 3-DES. 3-DES більш безпечний, але займає більше часу, коли шифрує файл, тобто дві опції, щоб дешифрувати це. Можна створити файл самодешифрування, який попросить пароль перед витяганням. MEO, проте, не пропонує стискування для файлів.

Інша робота шифрування, яку можна виконати з MEO, посилання зашифрованих листів, які не буде прочитано, якщо неможливо дешифрувати їх. У додатку є сервер SMTP, який дозволяє йому відправляти зашифровані повідомлення зсередини додатка, і він навіть підтримує приєднання (рис. 11.6).

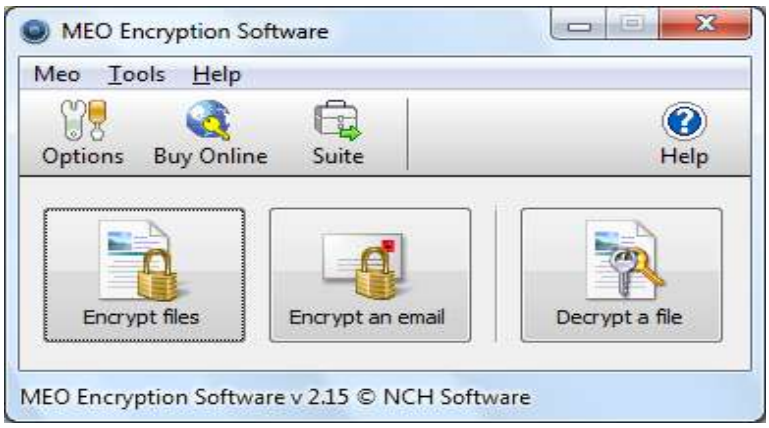


Рис. 11.6. **Головне вікно програми MEO Encryption Software**

Цікава особливість полягає в тому, що MEO працює і у Windows, і в Mac OS X, таким чином, можна спільно використовувати файли з тими платформами.

AxCrypt – це зручна і легка у використанні утиліта для шифрування будь-яких файлів. AxCrypt не додається у трей і не займає оперативну пам'ять. Ця програма після встановлення додається в контекстне меню файлів. Для того щоб захистити файл від сторонніх очей, досить клацнути на ньому правою кнопкою мишки і в меню, що відкрилося, у розділі AxCrypt, вибрати пункт "Шифрувати".

Буде запропоновано двічі ввести пароль (щоб не помилитися, інакше потім отримати файл назад не реально). Замість пункту " Шифрувати " можна вибрати " Шифрувати копію" або " Шифрувати копію .EXE". У першому випадку буде створено зашифровану копію файла, що відкривається програмою AxCrypt (оригінал залишиться незайманим), у другому – зашифровану копію файла, що відкривається на будь-якому комп'ютері без потреби в установленні програми (досить просто знати пароль) (рис. 11.7).

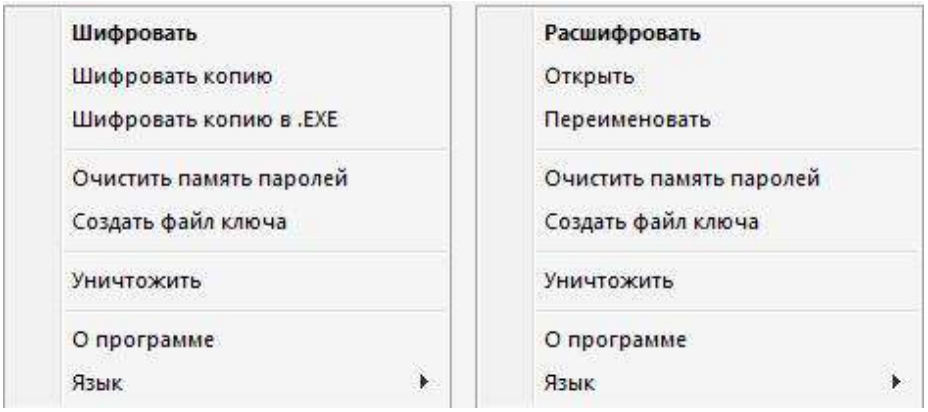


Рис. 11.7. **Діалогове вікно програми AxCrypt**

Алгоритми шифрування AES-128 і SHA-1. Після зашифровування у файла з'являється новий значок і розширення. Програма AxCrypt інтегрується у Windows і з'являється в контекстному меню, яке відкривається натисненням правої кнопки мишки, потім потрібно вибрати AxCrypt і натиснути "Зашифрувати". Інтерфейс AxCrypt мінімальний, підтримує 12 мов, серед яких російська. Але, ураховуючи простоту, програмою користуватися зручно й без цього.

Програма не займає багато оперативної пам'яті, не потребує особливих знань для користування. В опціях є такі можливості, як шифрування, розшифрування, редагування, стискування і переглядання файлів. AxCrypt має таку дуже корисну функцію, як захист ключем. Файл відкривається тільки тоді, коли в нього завантажать ключ, який може перебувати завжди при користувачі, разом з USB-носієм.

Encrypt Files – це програма шифрування файлів, вільно поширювана, щоб надійно зашифрувати та захищати конфіденційні дані. Це швидке і легке програмне забезпечення, установлене на комп'ютері, яке підтримує 13 просунутих алгоритмів шифрування, робить файли прихованими після шифрування. Encrypt Files дозволяє регулювати доступ до зашифрованих файлів (рис. 11.8).

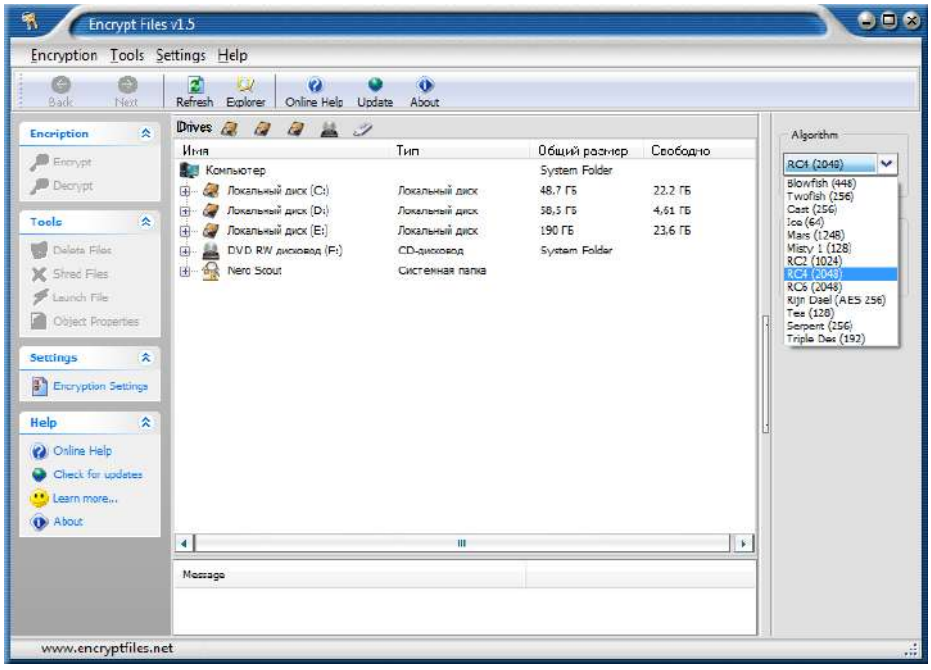


Рис. 11.8. **Головне вікно програми Encrypt Files**

Загальною особливістю всіх перелічених програмних продуктів є використання паролів (key), тобто ключовій послідовності, яка бере участь у побудові зашифрованої послідовності.

Пароль – це секретне слово, знання якого дає доступ до ресурсу, що захищається. Причому трактування поняття "ресурс" може бути вельми широким – від файлового архіву, зашифрованого за допомогою пароля, до комп'ютерів домену, доступ до яких користувач дістає, подавши правильний логін і пароль. Інструменти, про які буде розказано, може використовувати й хакер, що побажав дістатися до чужих таємниць, проте такі інструменти для нього здебільшого не найкращий варіант. Для користувача, що забув пароль, завдання зведено тільки до його отримання. Перед хакером, що цікавиться закритою інформацією, постає завдання дещо інше і ширше, бо, зазвичай, він шукає вразливість, знання якої дозволяє здобути контроль над чужим комп'ютером.

Паролі продуктів Microsoft Office

Наступним важливим завданням може стати відновлення паролів до документів Microsoft Office. У цьому разі необхідно скористатися програмним забезпеченням Advanced Office Password Recovery (рис. 11.9). Атаку на пароль здійснюють за допомогою словника, маски, а також прямим перебором (див. рис. 11.9).

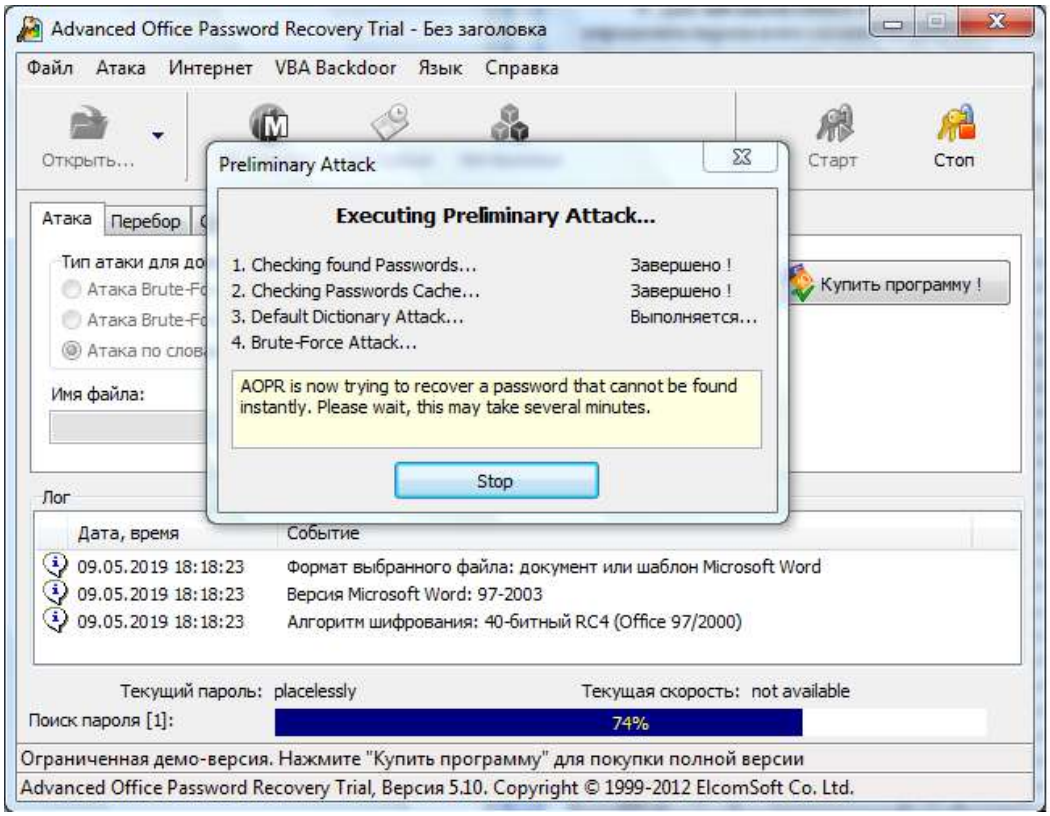


Рис. 11.9. **Атака прямим перебором у програмі Advanced Office Password Recovery**

Люди використовують деякі слова як паролі частіше за інші. Частотні словники перелічують найбільш популярні слова. Великі словники містять сотні тисяч слів. Великий частотний словник містить назви корпорацій, заголовків кінофільмів, торговельні марки тощо. Злом за допомогою словника, зазвичай, відбувається дуже швидко, навіть якщо використовують величезний словник. У пакет PwlTool не вміщено словників.

Можна використовувати відносно малі словники (~80 КБ) від webdon.com/DownLoad/dic1.zip або величезний (~9 МБ) – www.kull.ch/Bauersachs/download/allwords2.zip. У версії v6.0 є гібридне дослідження. Цей метод корисний для злому паролів типу john43. Коли перемикач "Гібридне дослідження" включено, RePwl буде пробувати всі варіанти, подібно до wordXXX, де word – це слово від словника і XXX – суфікс, сформований згідно з параметрами налаштування рішення "атака в лоб".

Важливо! Упевніться, що всі слова у словнику перебувають у верхньому регістрі.

"Атака в лоб" (пошук усіх можливих паролів) не підходить для довгих паролів, тому що потрібно дуже багато часу. Головним чином є комбінації, подібно до jkqmzwd, які є повністю безглуздими серед мільярдів і квінтильйонів розшукуваних паролів. Посилена "атака в лоб" – це оптимізований алгоритм дослідження, який тільки пробує "розумні" паролі. У результаті час злому скорочується.

Але це також має деякі недоліки:

поточна версія підтримує тільки англійську мову;

посилена "атака в лоб" не відновлює паролі, які містять цифри або символи. Наприклад, пароль soft4you не відновлюється із застосуванням цього алгоритму;

деякі слова важкі для такого алгоритму атаки. Потрібно визначити рівень як ціле число в діапазоні 1 ... 26. Розумні значення для цього типу атаки – 9 ... 16 (значення за замовчуванням – 13).

Під час спроби відновлення паролів на відкриття документа від Office 2007 і вище варто врахувати, що в цьому разі для шифрування документів застосовано алгоритм AES із довжиною ключа 128 розрядів, а також гешування за алгоритмом SHA-1 (паролі до Word, Excel, PowerPoint, Access). Тільки прості й короткі паролі може бути виявлено простим перебором. Для прискорення процесу відновлення паролів рекомендовано застосовувати сучасні графічні карти від nVidia та AMD/ATI (рис. 11.10).

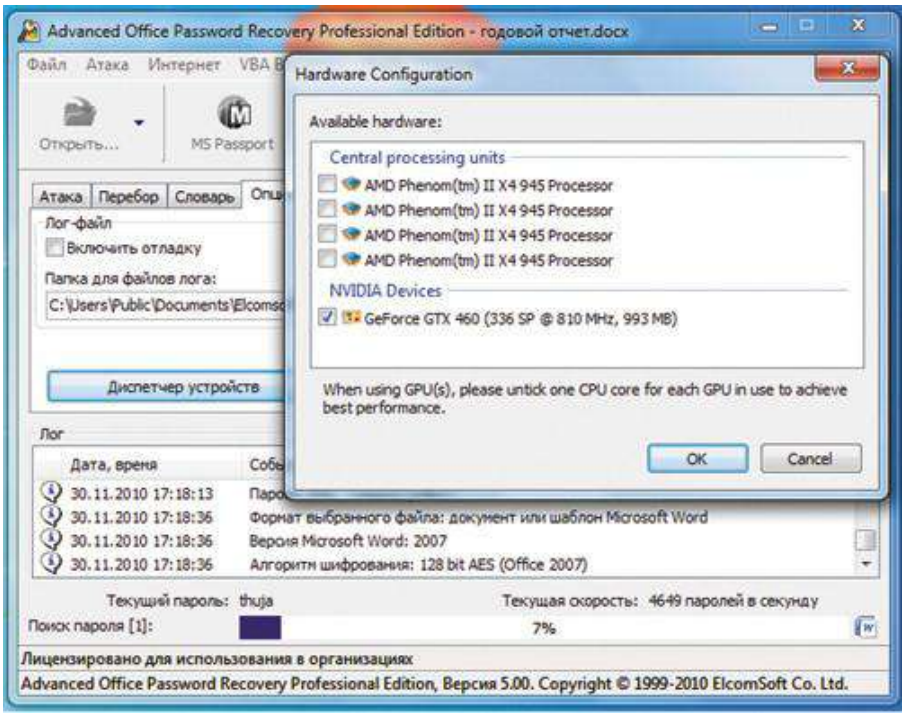


Рис. 11.10. **Відновлення пароля до документа Office 2007 з використанням відеокарти**

Проте набагато частіше потрібно знати не сам пароль до того або того документа, а вміст документа. Для видалення пароля з документів Office призначено програмне забезпечення **Advanced Office Password Breaker**.

Видалення парольного захисту можливе з файлів, збережених у форматі .doc і .xls. Головне вікно Advanced Office Password Breaker показано на рис. 11.11.

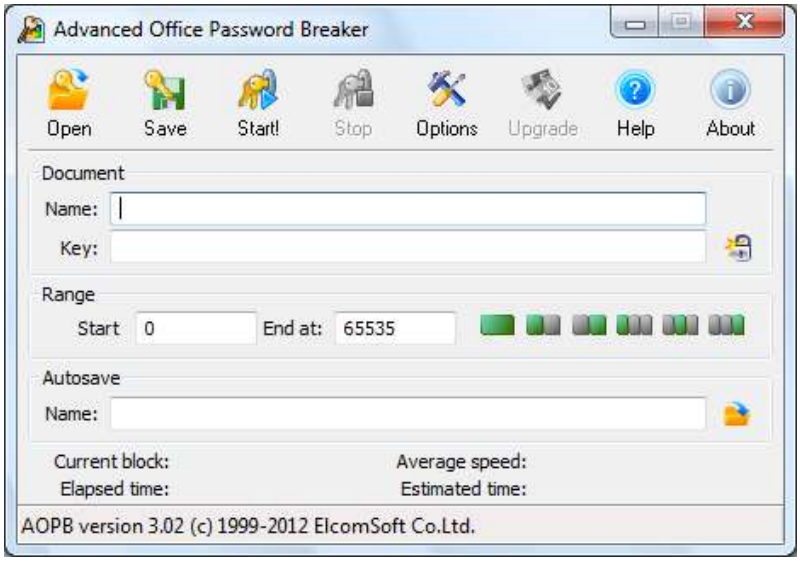


Рис. 11.11. **Видалення пароля документа Word у програмі Advanced Office Password Breaker**

Перебір 40-розрядних ключів можна здійснити за заздалегідь відомий проміжок часу, залежний від потужності процесора і кількості ядер. Час знаходження ключа становить близько 5 днів для одноядерного процесора. У разі використання процесорів Core 2 Duo і Core 2 Quad час пошуку ключа зменшується пропорційно до кількості ядер. Продукт Advanced Office Password Breaker Professional підтримує до 4 процесорів або ядер, дозволяючи прискорити процес перебирання ключів у 4 рази. Корпоративна версія підтримує до 32 процесорів.

Захист документів Microsoft Word і Microsoft Excel далеко не завжди дозволяє швидко знайти забутий пароль. Використовуючи атаку за словником, прямий перебір паролів та інші методи, відновити пароль можна, проте цей процес може забрати багато часу.

Разом із тим, у разі якщо захист документа сумісний із форматом Microsoft Office 97, є рішення, що дозволяє гарантовано відкрити документ протягом невеликого проміжку часу. Цей формат захисту вразливий, оскільки в ньому використовують ключ шифрування довжиною всього 40 розрядів.

Для прискорення процесу перебирання ключів шифрування фірма Elcomsoft розробила технологію використання заздалегідь обчислених таблиць (Thunder Tables). Використовуючи ці таблиці, можна знайти ключ шифрування для документа Word версій 97 – 2000 всього за декілька хвилин. Якщо необхідний пароль до Microsoft Excel 97 – 2000, використовують класичні веселкові таблиці. Вірогідність розшифрування документа за декілька хвилин становить 97 %.

Програми підбору паролів до продуктів Microsoft Office можна скачати зі спеціалізованих сайтів, наприклад, таких як:

http://www.passwords.ru – сайт комплексних рішень захисту файлів;

http://www.elcomsoft.com – сайт рішень компанії Elcomsoft;

http://lastbit.com/mso/default.asp – сайт Password Recovery Solutions.

На сайті комплексних рішень захисту файлів розміщено посилання на ресурси, що належать до теми паролів. Для прикладу наведено дві програми з досить великого списку: http://www.passwords.ru/aopb.html Advanced Office Password Breaker (AOPB) – це програма розшифрування документів Word і Excel, захищених паролем на відкриття документа.

Забезпечує гарантоване відкриття документа, незалежно від складності пароля та його довжини, що досягають шляхом перебирання 40-бітових ключів шифрування. Таку невелику довжину ключа використовують у Microsoft Office із-за експортних обмежень.

Advanced Office Password Recovery (AOPR) – це програма для відновлення втрачених паролів до документів Microsoft Office. Поставляють її у двох редакціях – Standard і Professional. Більшість паролів знаходять миттєво прямим декодуванням. Інтерфейс російський і англійський (рис. 11.12).

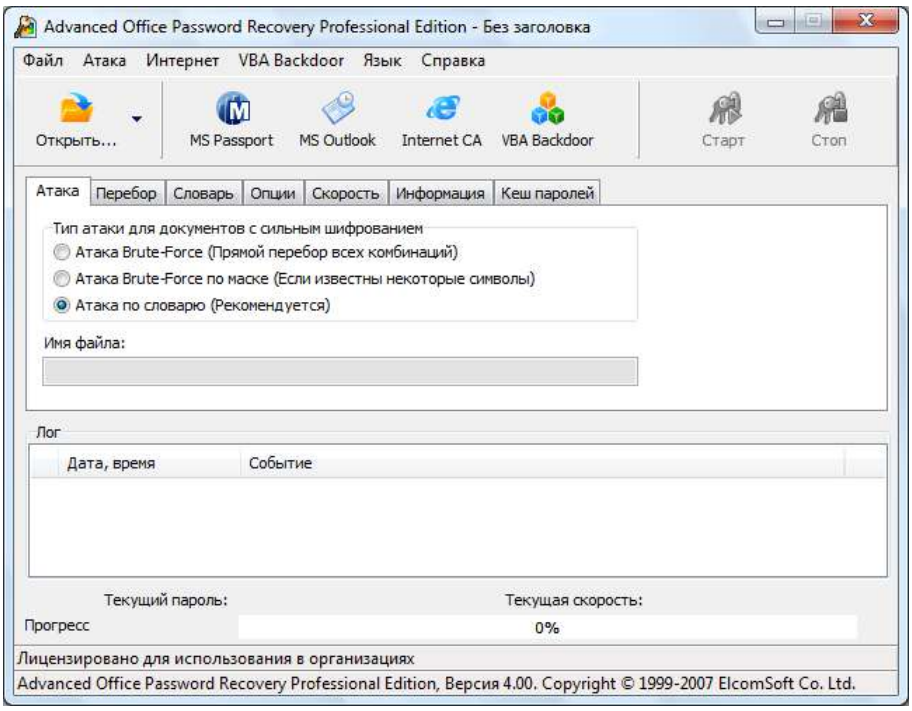


Рис. 11.12. **Вікно програми Advanced Office Password Recovery**

Паролі, що захищають архівні файли та файли з документами різних програмних продуктів, переважно, не зберігають у цих документах у зашифрованому вигляді. Пароль використовують як ключ шифрування і ніде не зберігають.

Під час використання утиліти Advanced Office Password Recovery можливе відновлення паролів усіх документів. Незважаючи на те що MS Word, шифруючи документ, використовує досить складний алгоритм (пароль не зберігають усередині файла), його може бути розкрито з використанням методу "грубої сили" або атаки за словником. Середня швидкість під час використання процесора Intel становить 1 млн паролів на секунду.

В Excel використовують ті самі технології захисту, що й Word. Відмінність становлять тільки паролі на книги/листи. Геш-кодування паролів зберігають у документі. Найпарадоксальніше, що довжина геш-кодування всього 16 бітів. Отже, для кожного геш-кодування є безліч відповідних паролів. Наприклад, можна захистити лист паролем test і спробувати його відкрити за допомогою пароля zzyw. Відмінність пароля захисту книги

від пароля листа полягає в тому, що документ шифрується. Паролі зберігають у заголовку файла. Заголовок зашифровано за RC4, але ключ шифрування має довжину 32 біти, його зберігають в одній із системних DLL. Знаючи цей ключ, можна знайти будь-який пароль на базу Access.

Паролі продуктів Adore Reader

Для злому використовують програмний продукт **Advanced PDF Password Recovery.** Ця версія підтримує роботу з документами, пароль на яких було встановлено в Adobe Acrobat, починаючи з версії 3.x і закінчуючи 9.x, також можна дістати підтримку роботи зі 128-bit RC4 шифруванням і 256-bit AES шифруванням, без проблем зняти все наявні обмеження з документа формату PDF, зробити розшифрування документа, якщо пароль відкритий і відомий, також вести атаку за повним перебиранням на відкритий пароль, Advanced PDF Password Recovery вміє атакувати на відкритий пароль, використовуючи для цього словники.

Розробники гарантують відновлення пароля, який використовував 40-бітове шифрування, і цей процес забере декілька днів, якщо використовувати багатоядерні процесори, то результат буде швидшим, також можна прискорити процес, якщо вбудовано GPU-прискорення для відеокарт NVIDIA (рис. 11.13).

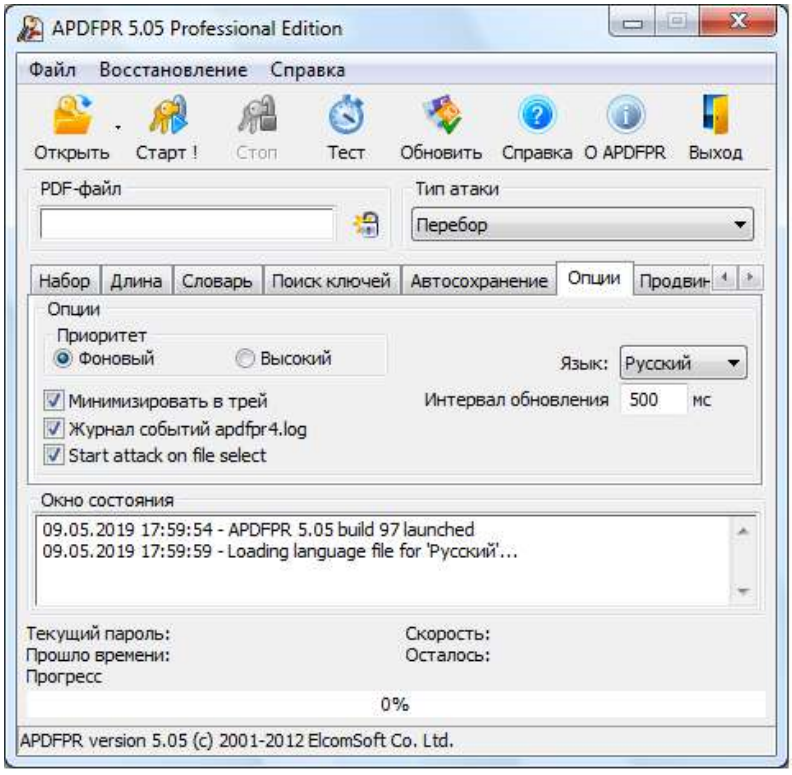


Рис. 11.13. **Вікно програми Advanced PDF Password Recovery**

Advanced PDF Password Recovery гарантовано відновить пароль 40-розрядного шифрування з використанням технології Thunder Tables за декілька хвилин.

Паролі архіваторів

На сьогодні є досить багато алгоритмів криптографічного захисту інформації в архіваторах. Проте в переважній більшості архіваторів реалізовано, зазвичай, який-небудь один метод. Мова йде про найбільш поширені архіватори. Водночас необхідно визнати, що на сьогодні ZIP-кодування, як і шифрування за алгоритмом DES (Data Encryption Standard), складно назвати стійким. Найбільш надійним зараз є алгоритм AES, узятий як стандарт у США 2001 року. Якщо подивитися на найпопулярніші в Україні архіватори, то це, без сумніву, WinZip і WinRar. Слід розглянути шифрування в цих архіваторах більш докладно.

В архіваторі **WinZip** реалізовано три алгоритми шифрування:

Standard Zip 2.0 encryption – його використовують за замовчуванням;

128-bit AES encryption – це криптографічний алгоритм AES із довжиною ключа 128 розрядів;

256-bit AES encryption – це криптографічний алгоритм AES із довжиною ключа 256 розрядів (посилений алгоритм шифрування).

На жаль, переважно, користувачі застосовують перший алгоритм (за замовчуванням), адже він є найбільш слабким. Адже, якщо не знаєте, за допомогою якого архіватора отримувач архіву буде розпаковувати його, то змушені задіювати метод за замовчуванням.

Разом із тим необхідно підкреслити, що який би алгоритм шифрування не використали, стійкість шифру, насамперед, буде залежати від стійкості ключа. Отже, під час використанні шифрування необхідно застосовувати стійкі паролі.

До недоліків шифрування WinZip варто також зарахувати те, що WinZip не шифрує коментарі zip-файлів і такі властивості зашифрованих файлів, як назви, дати тощо. Адже це вельми цінна інформація для аналітика. Далеко не всякий користувач перейменовує файли, що архівуються, а вже тим більш змінює решту атрибутів (дата створення, зміни, розмір тощо).

В архіваторі **WinRar** застосовують алгоритм шифрування AES із довжиною ключа 128 розрядів. Варто зазначити, що файли, зашифровані у WinRar, будуть відкриватися і у WinZip. Проте зворотне буде правильним лише для алгоритму шифрування Zip 2.0. Крім того, варто запам'ятати, що якщо ви вирішили запакувати деякі дані в архів із шифруванням, необхідно поклопотатися про надійне видалення файлів із носія, на якому вони перебували.

Advanced Archive Password Recovery – це програмне забезпечення призначене для відновлення паролів до архівів zip, rar, ace, arj (рис. 11.14).

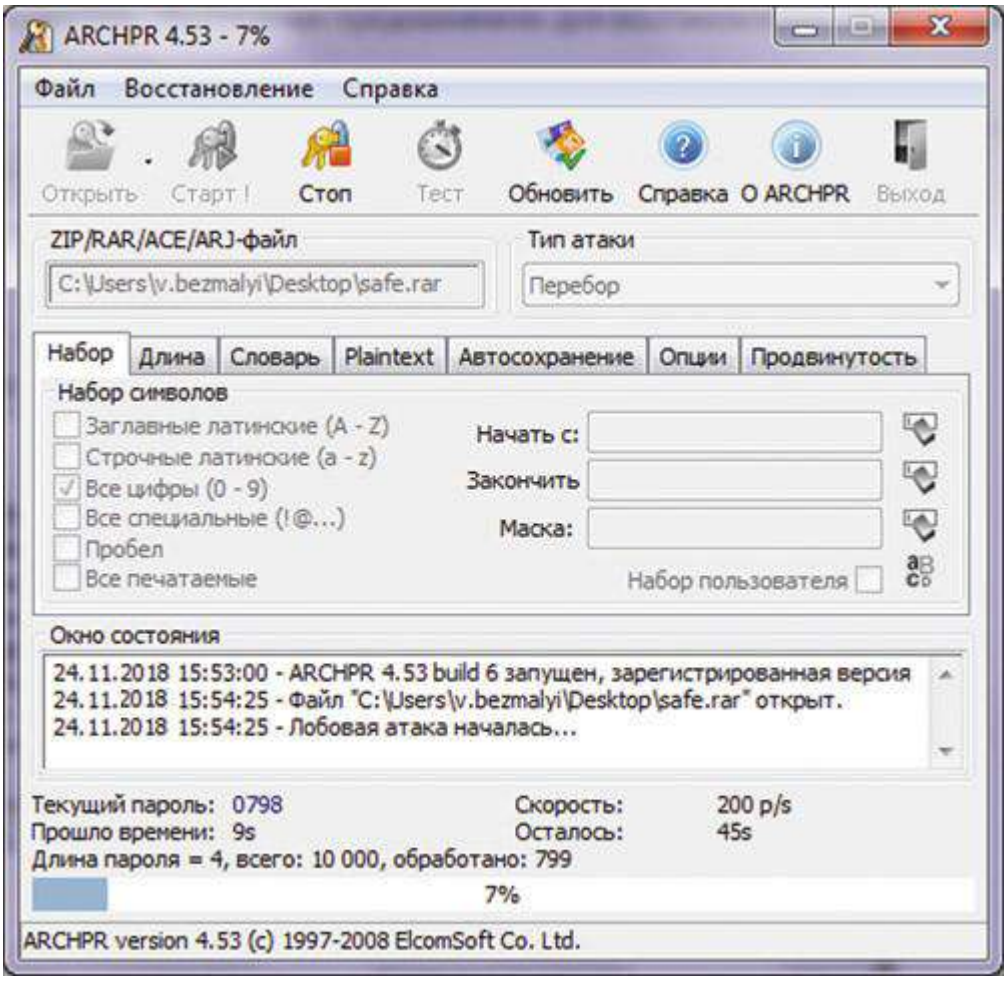


Рис. 11.14. **Вікно програми Advanced Archive Password Recovery**

Відновлення пароля до архіву RAR. Це програмне забезпечення відновлює доступ до зашифрованих архівів двома шляхами:

знімає парольний захист;

відновлює оригінальний текстовий пароль.

Підтримувані формати архівів – ZIP/PKZip/WinZip, RAR/WinRAR, ARJ/WinARJ, а також ACE/WinACE (1.x), створені будь-якими програмами-архіваторами, і архіви, що саморозпаковуються, створені у PKZip, WinZip, RAR і WINRAR.

Під час використання WinZip 8.0 і більш молодших версій гарантовано зняття захисту протягом години. Крім того, варто зазначити, що за наявності хоча б одного файла з архіву весь архів буде розшифровано за хвилину (для архівів у форматі ZIP і ARJ).

Виконання крекінгу захисту виконуваного файла OllyDbg

Для виконання крекінгу виконуваного файла буде потрібно:

1. *Наладник.* Саме за допомогою цієї програми буде здійснено аналіз виконуваної коди тестової програми. Для прикладу вибрати наладник **OllyDbg v.2.01.**

2. *Hex-редактор.* За допомогою цієї програми буде виконано модифікацію початкової коди. Для прикладу вибрати програму **Xvi32.**

3. *Тестова програма* **TESTP.exe,** яку необхідно крекнути.

Увесь указаний софт не потребує інсталяції. Спершу запускається наладник OllyDbg. Для роботи знадобиться всього три вікна:

1. CPU.
2. Breakpoints.
3. Patches.

Необхідно зробити так, щоб програма приймала будь-який ключ і виводила повідомлення про реєстрацію, незалежно від уведеного значення. Початкове вікно після запуску програми **TESTP.exe** має такий вигляд (рис. 11.15):

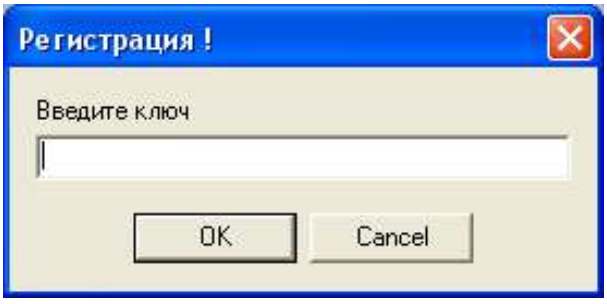


Рис. 11.15. **Діалогове вікно тестової програми TESTP**

1. Відкрийте програму **TESTP.exe** в наладнику OllyDbg. Для цього відкривайте меню **File** → **Open** (F3) і виберіть тестову програму. Після завантаження у вікні CPU буде такий лістинг (рис. 11.16):

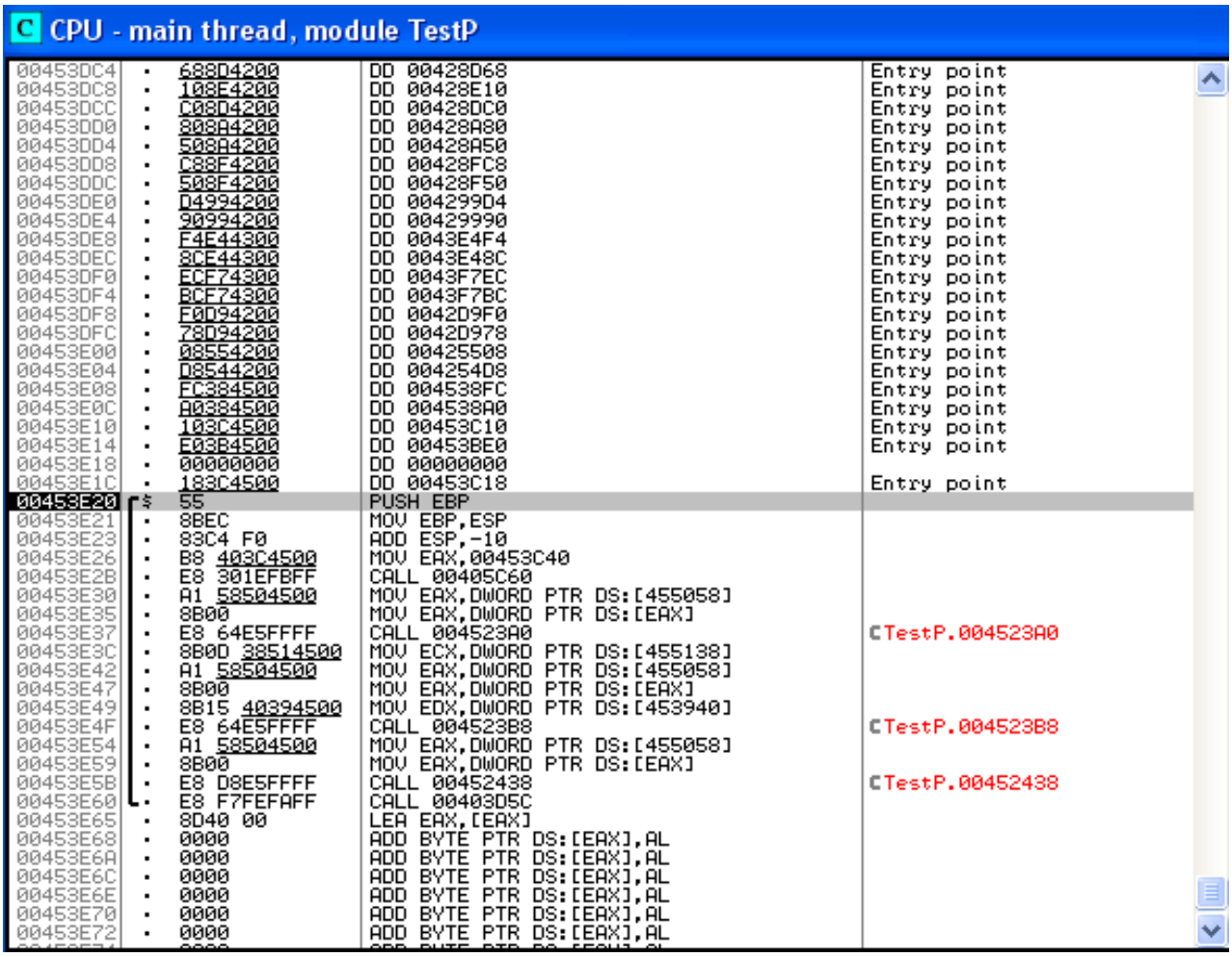


Рис. 11.16. **Головне вікно програми OllyDbg із завантаженим testp.exe**

Це і є асемблерний код тестової програми, наприклад:

`PUSH EBP ;` Початок іншої функції;

`CALL TestP.00405C60 ;` Виклик функції.

2. Тепер потрібно знайти функцію, яка видає вікно із запитом уведення ключа. Для цього виконують програму покроково, натискаючи клавішу F8 до тих пір, поки не з'явитися вікно із запитом уведення. Після кількох натиснень з'являється діалогове вікно введення (рис. 11.17):

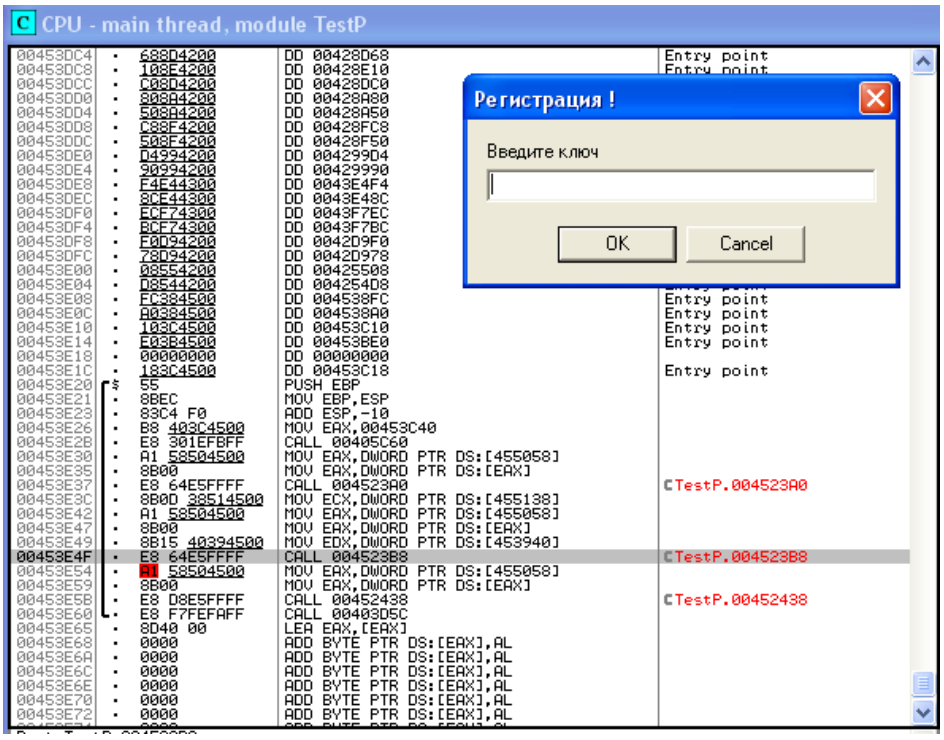


Рис. 11.17. **Пошук першої точки зупину**

Тепер відомо, що десь у цій функції (TestP.004523B8) виводиться діалогове вікно введення ключа. Необхідно докопатися до функції, яка відображає це вікно. Тому треба зайти в цю функцію. Тепер перед цим рядком `CALL TestP.004523B8` слід поставити точку зупину (Breakpoint), для цього виділити рядок перед нею, натиснути **F2** – і у віконці Breakpoints з'явиться такий рядок (рис. 11.18):

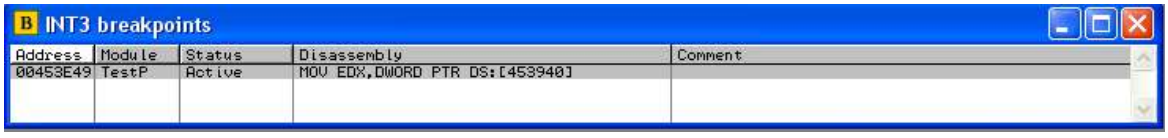


Рис. 11.18. **Вікно Breakpoints програми OllyDbg**

Далі виконати рестарт програми **Debug** → **Restart** (Ctrl + F2) для того, щоб завантажити тестову програму наново. Після чого потрібно дійти до точки зупину. Для цього натиснути ▶, щоб виконати програму. У результаті покрокове виконання програми зупиниться на першій точці зупину.

Для того щоб зайти у функцію натиснути клавішу F7. Відмінність клавіш очевидна: F8 – трасує програму без заходу у функції, F7 – із заходом у функції. Можна бачити таку картину (рис. 11.19):

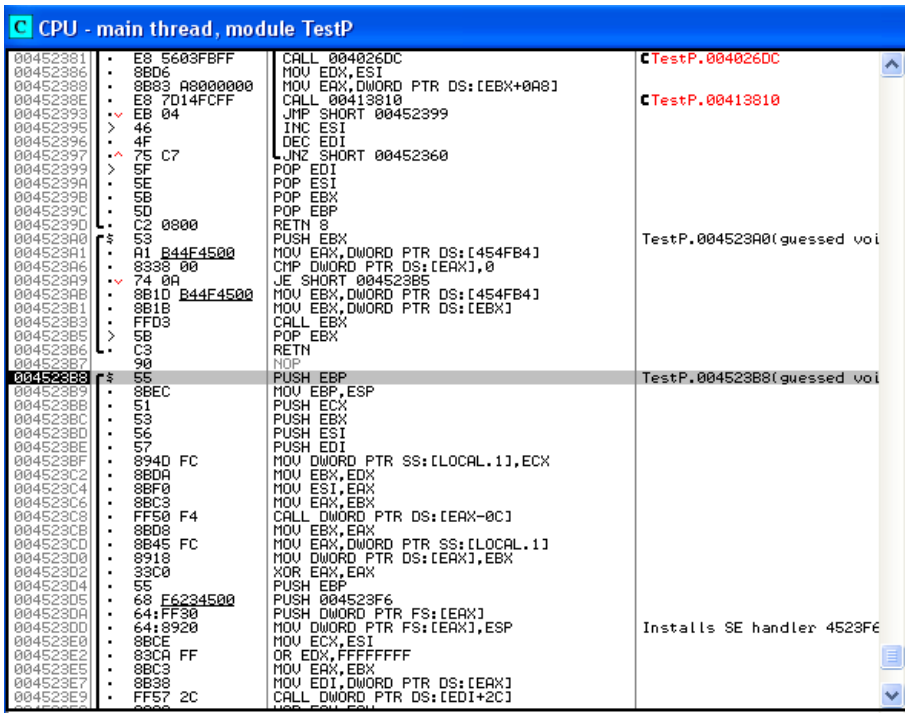


Рис. 11.19. **Виконання програми до першої точки зупину**

Треба продовжувати, доки не знаходять функцію, яка виводить вікно, обробляє натиснення на кнопку OK і визначає, чи правильно введено ключ, чи ні. Продовжуючи, також натискають F8 поки не з'явитися вікно (рис. 11.20):

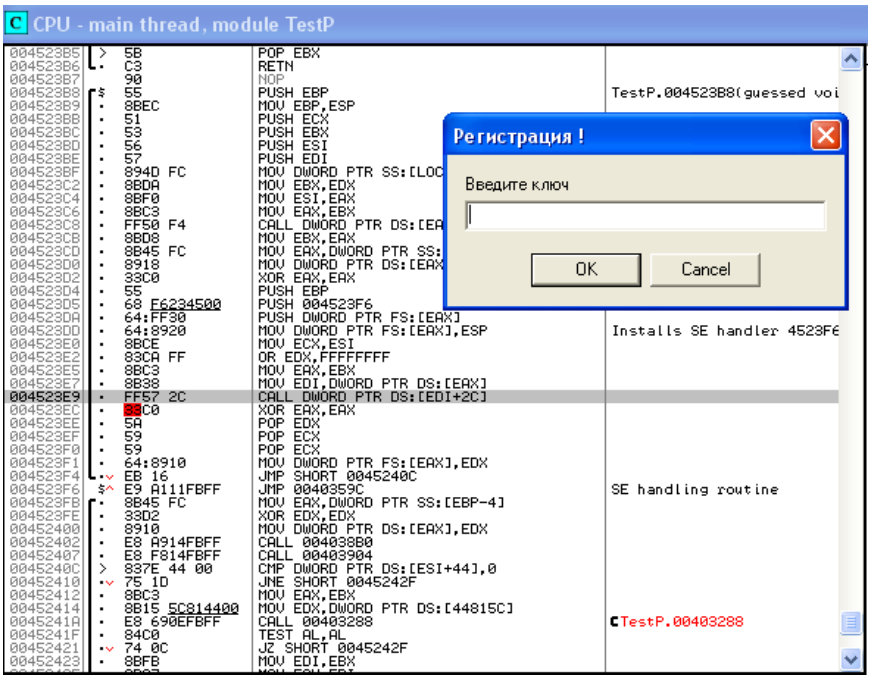


Рис. 11.20. **Пошук другої точки зупину**

Знову перед нею ставлять другу точку зупину. Натискають Ctrl + F2 і знову, бачачи що виконання зупинилося на першій точці зупину, а потрібно до другої, для цього натискають ще раз. Виконання дійде до другої точки зупину. Зайшовши у функцію клавішею F7, можна бачити таке (рис. 11.21):

Рис. 11.21. **Виконання програми до другої точки зупину**

Продовжуючи далі, знаходять чергову функцію. Знову ставлять третю точку зупину і, повертаючись сюди, заходять у функцію. Продовжують шукати, натискаючи F8. Під час натиснення F8 бачать, що починають з'являтися API-функції (GetCapture, GetActiveWindow). Це означає, що вже близько заповітне діалогове вікно введення ключа (рис. 11.22):

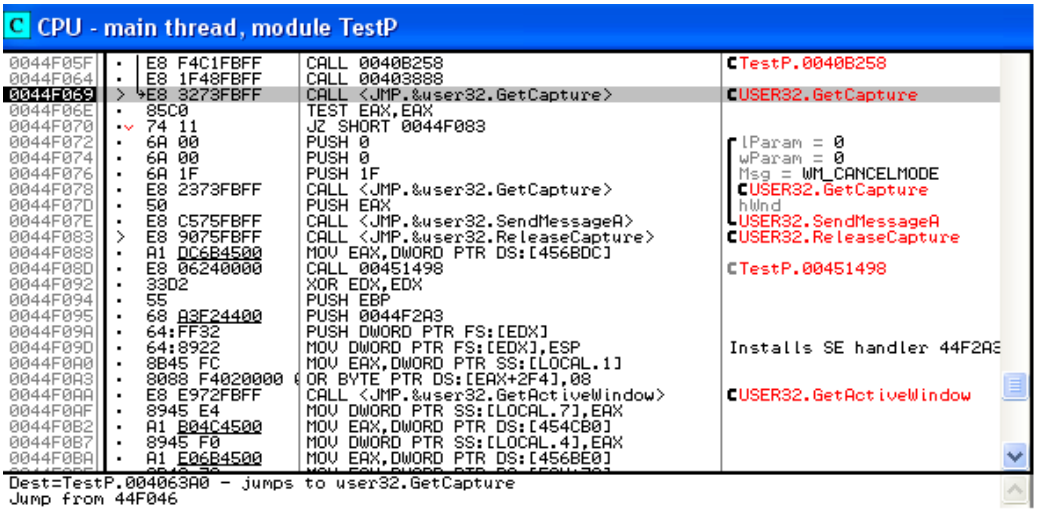


Рис. 11.22. **Пошук третьої точки зупину**

Натискають поволі та стежать, коли з'явитися вікно на панелі завдань під час трасування програми (рис. 11.23), яке з'явилося на цьому рядку коду (рис. 11.24).

Рис. 11.23. **Стан панелі завдань під час трасування**

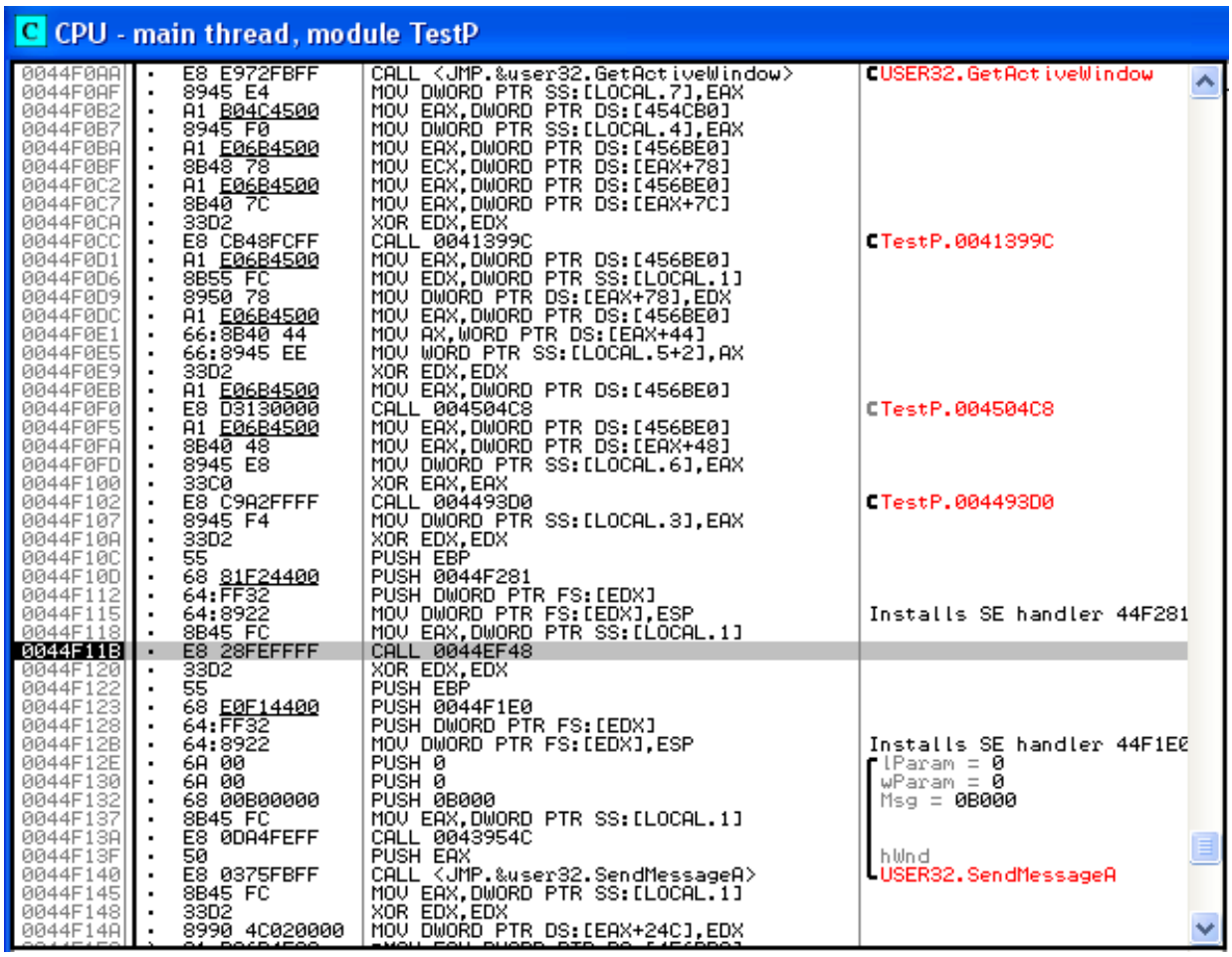


Рис. 11.24. **Виконання програми до третьої точки зупину**

Під час натиснення на нього мишкою вікно ще не відображається. Саме ця функція рисує вікно, і десь тут звіряється введений користувачем ключ. Продовжуючи далі трасувати програму (F8), бачать, що на цьому місці якийсь нескінченний цикл. Ставлять чергову точку зупину на першому рядку після циклу (`MOV DWORD PTR SS:[EBP-8],EAX`) (рис. 11.25):

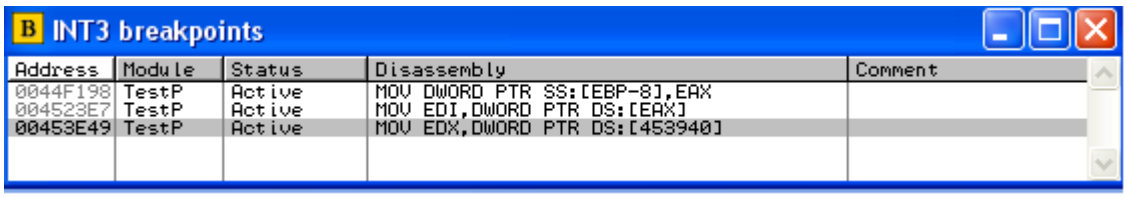


Рис. 11.25. **Поточний стан вікна Breakpoints програми OllyDbg**

Натискаючи кнопку  , бачать, що віконце повністю намальовано, чекає введення, а програма зупинилася тут, у циклі. Далі вводять будь-який ключ, натискають ОК (рис. 11.26).

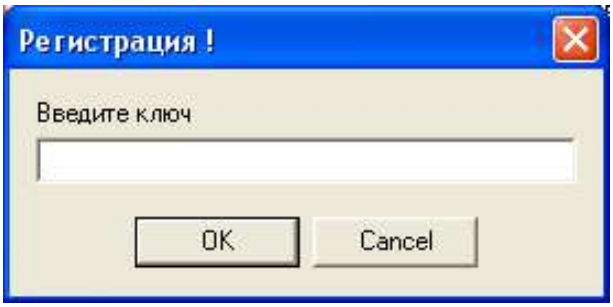


Рис. 11.26. **Тестове введення довільного ключа**

Віконце закривається і відбувається повернення у вікно наладника, а виконання зупиняється на останній точці зупину. Тепер потрібно знайти функцію, у якій виводиться вікно з повідомленням, що ключ уведено неправильно. Для цього трасують (F8) цю програму до того моменту, доки не з'явиться необхідне вікно з повідомленням (рис.11.27).

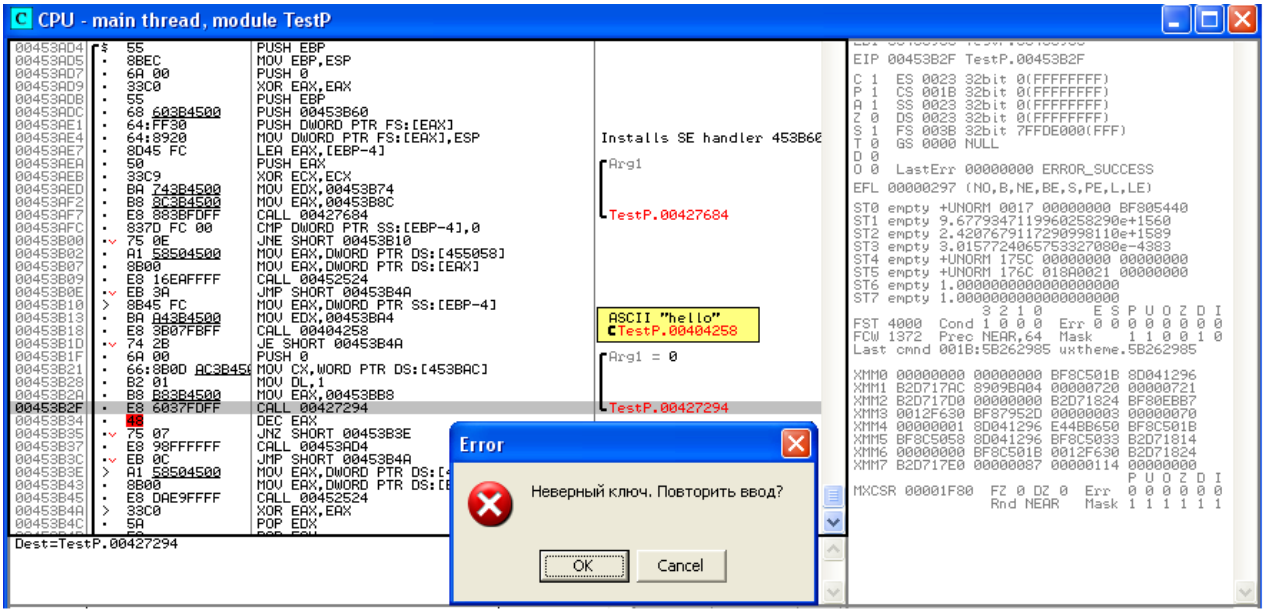


Рис. 11.27. **Вхід до програми порівняння пароля з еталоном**

У лістингу видно рядок ASCII hello: це і є **правильний ключ.**

Наступним завданням буде необхідність зробити так, щоб програма приймала будь-який ключ. Слід проаналізувати код перед викликом функції (`CALL TestP.00427294`), яка виводить вікно з повідомленням про помилку. Програма має порівняти введений ключ із яким-небудь іншим або перевірити його правильність будь-яким іншим способом. Але для цього

вона має викликати функцію, яка це все зробить. Слід шукати приблизно таку структуру:

MOV EAX ... // записує в EAX адресу цього рядка;

CALL ... // виклик функції, яка перевірить правильність цієї функції;

JNZ (або JE) ... // умовний перехід.

Дещо про умовний перехід. За допомогою цього оператора можна переходити до певного рядка коду, адресу вказують після оператора. Отже, після виклику функції, якщо ключ було введено правильно, то після виклику цього оператора він перейде на рядок, після якого й запуститься правильне виконання програми TESTP.exe. Якщо ж ні, то буде виконуватися код після нього.

Під час аналізу коду видно таку структуру (рис. 11.28):

MOV EAX,DWORD PTR SS:[EBP-4];

MOV EDX,TestP.00453BA4;

CALL TestP.00404258;

JE SHORT TestP.00453B4A.

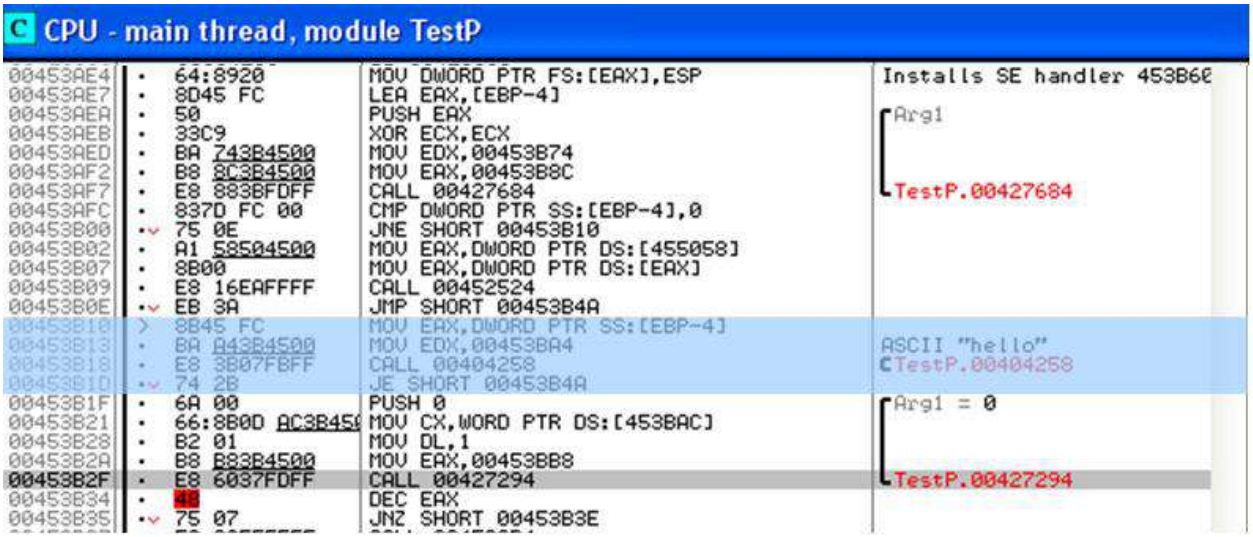


Рис. 11.28. **Функція перевірки пароля**

Тепер наново потрібно покроково виконати цю частину коду і стане зрозумілим, що цей оператор `JE SHORT TestP.00453B4A` пропускається й нікуди не перекидає. Отже, це і є функція перевірки правильності введеного значення ключа, і якщо він виявився неправильним, то оператор **JE** нікуди не перевів.

Отже, якщо код буде введено правильно, то цей оператор перекидає за адресою 00453B4A і все буде ОК. Подивіться, де ця адреса знаходиться (виділено рядок сірим кольором) (рис. 11.29):

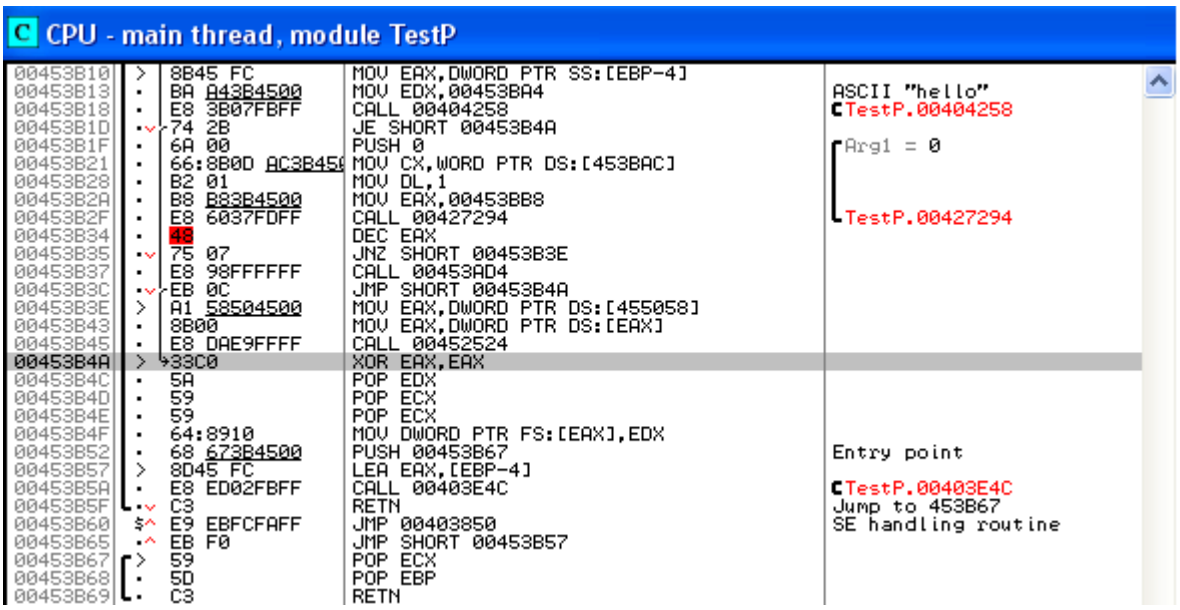


Рис. 11.29. **Визначення адреси переходу за правильним паролем**

Видно, що виклик функції, яка виводить повідомлення про те, що ключ неправильний, пропускається.

Замінити цього оператора на оператора безумовного переходу (**JMP**), який у будь-якому разі перекидає нас за цією адресою. Для того щоб замінити рядок, виділіть його, натисніть пропуск (Space) і замініть його (рис. 11.30):

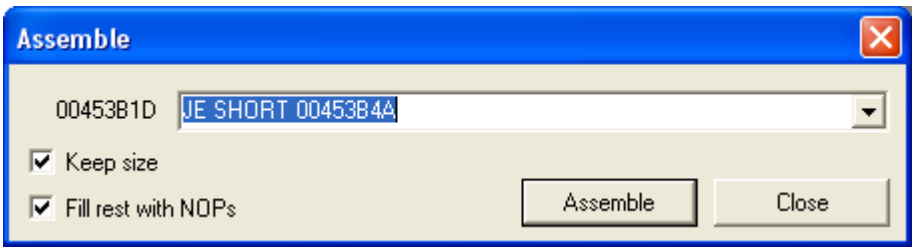


Рис. 11.30. **Вікно заміни команди умовного переходу**

на такий (рис. 11.31):

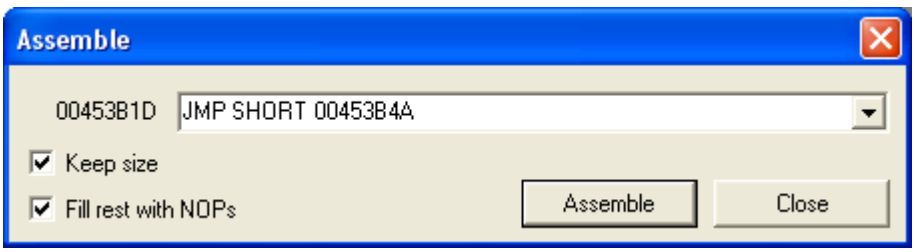


Рис. 11.31. **Вікно введення команди безумовного переходу**

Видно, що рядок **JE SHORT TestP.00453B4C** замінився і у віконці **Patches** додався рядок (рис. 11.32):

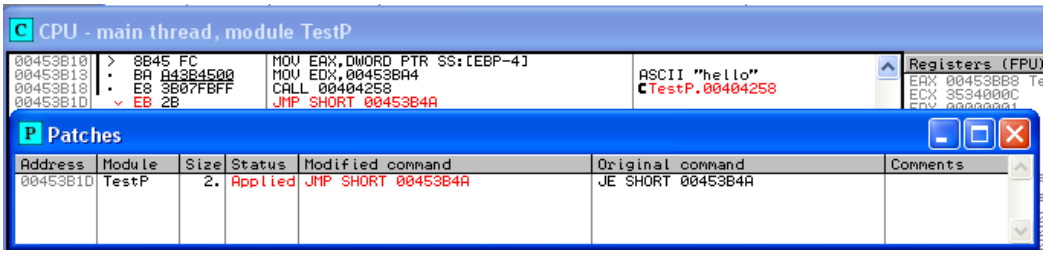


Рис. 11.32. **Результат заміни команди JE на команду JMP**

Конкретно ця програма завжди буде питати ключ, оскільки це просто тест, а реальна програма записала б собі в ініціалізації, що користувач правильно її зареєстрував.

Примітка. Слід ураховувати, що пошук необхідного шматочка коду не завжди буде таким простим. Іноді доведеться пробувати по черзі багато інших структур такого типу.

3. Останній крок, який треба зробити, – створити Patch для програми. Із цією метою буде задіяно Hex-редактор **Xvi32**. У ньому відкривають тестову програму TESTP.exe. Не слід лякатися великої кількості шістнадцяткових цифр. Адже файл розкладено побайтно.

Потрібно знайти рядок JE SHORT TestP.00453B4A в шістнадцятковому вигляді та позначити на JMP SHORT TestP.00453B4A. У вікні CPU (OllyDbg) навпроти кожного рядка написано її шістнадцятковий вигляд. Для усунення помилки слід узяти одну-дві попередні команди для пошуку, тобто не просто шукати **74 2B**, а вибрати ще попередній рядок коду і шукати так: **E8 3B 07 FB FF 74 2B.** Просто такий маленький рядок, як 74 2B, може повторюватися кілька разів, тому вибирають ще попередній рядок, щоб знайти саме те, що потрібно (рис. 11.33):

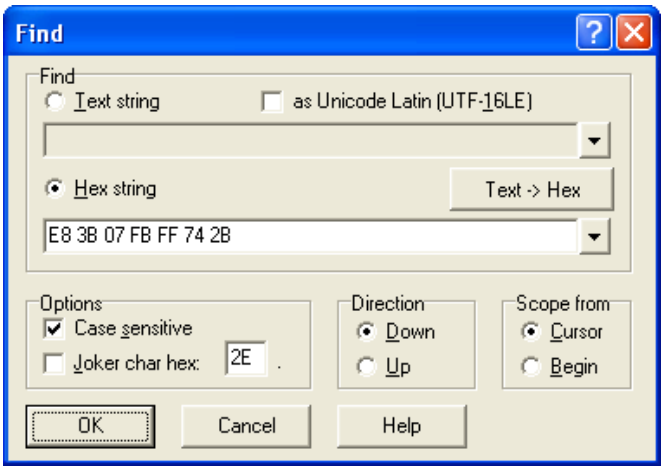


Рис. 11.33. **Головне вікно програми Xvi32**

Далі дивляться, якого у шістнадцятковому коді вигляду набуває видозмінений рядок **JMP SHORT TestP.00453B4A** (вікно CPU в OllyDbg). Він має такий вигляд: **EB 2B.** Слід замінити знайдений рядок на новий, для цього натискають кнопочку Replace All і замінюють рядки так (рис. 11.34):

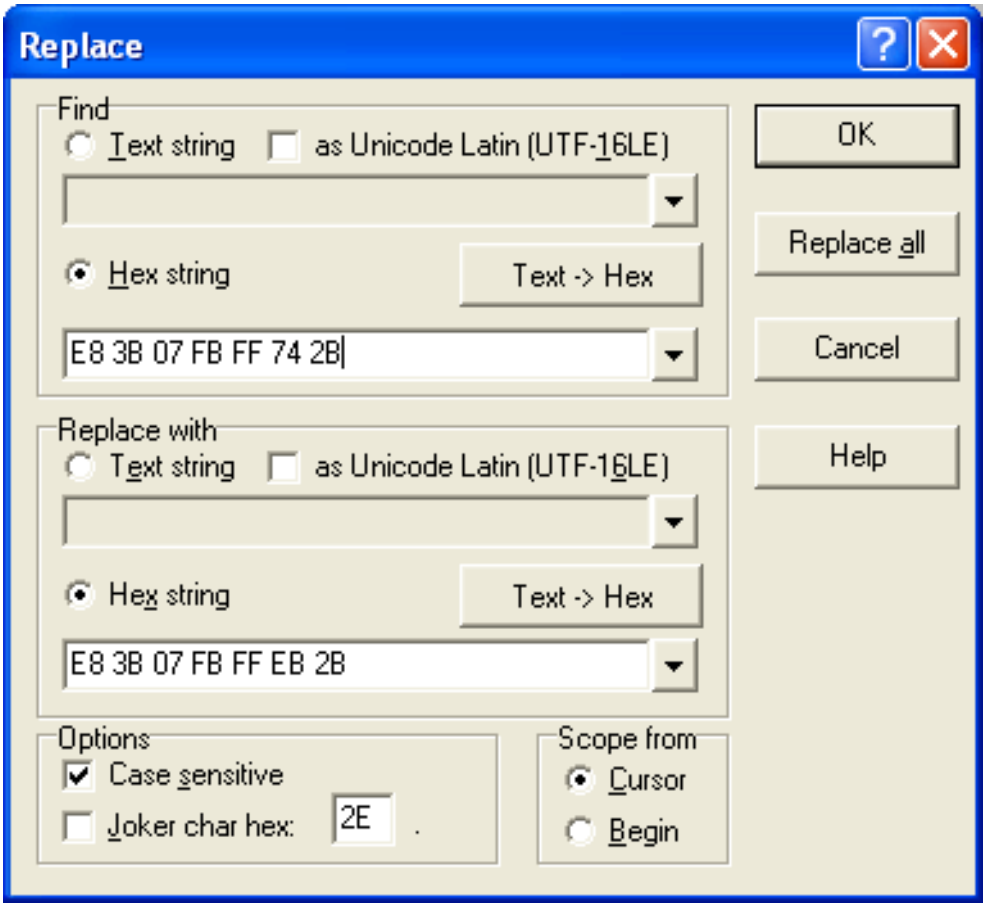


Рис. 11.34. **Створення коректного Patch у програмі Xvi32**

Зберігаючи зміни, закривають редактор. От і все, ця програма буде приймати будь-який ключ, що й потрібно було зробити.

Установлення і налаштування криптопровайдера Vipnet CSP

Отримання і встановлення ViPNet CSP

1. Для отримання ViPNet CSP необхідно перейти на офіційний сайт розробника за адресою https://infotecs.ru/downloads/besplatnye-produkty/

vipnet-csp.html і вибрати дистрибутив ViPNet CSP, відповідний операційній системі (рис. 11.35).

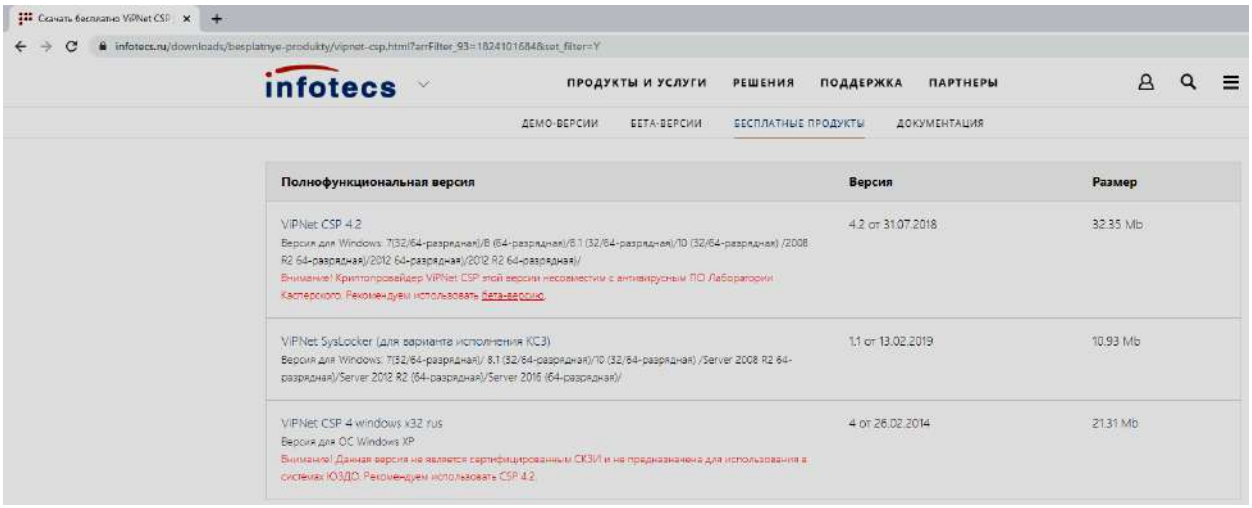


Рис. 11.35. **Запит на скачування дистрибутиву VipNet CSP**

2. Пройдіть встановлену процедуру реєстрації, згідно з умовами ліцензійної угоди (EULA), і заповніть обов'язкові поля (приклад на рис. 11.36, позиції 1 – 4).

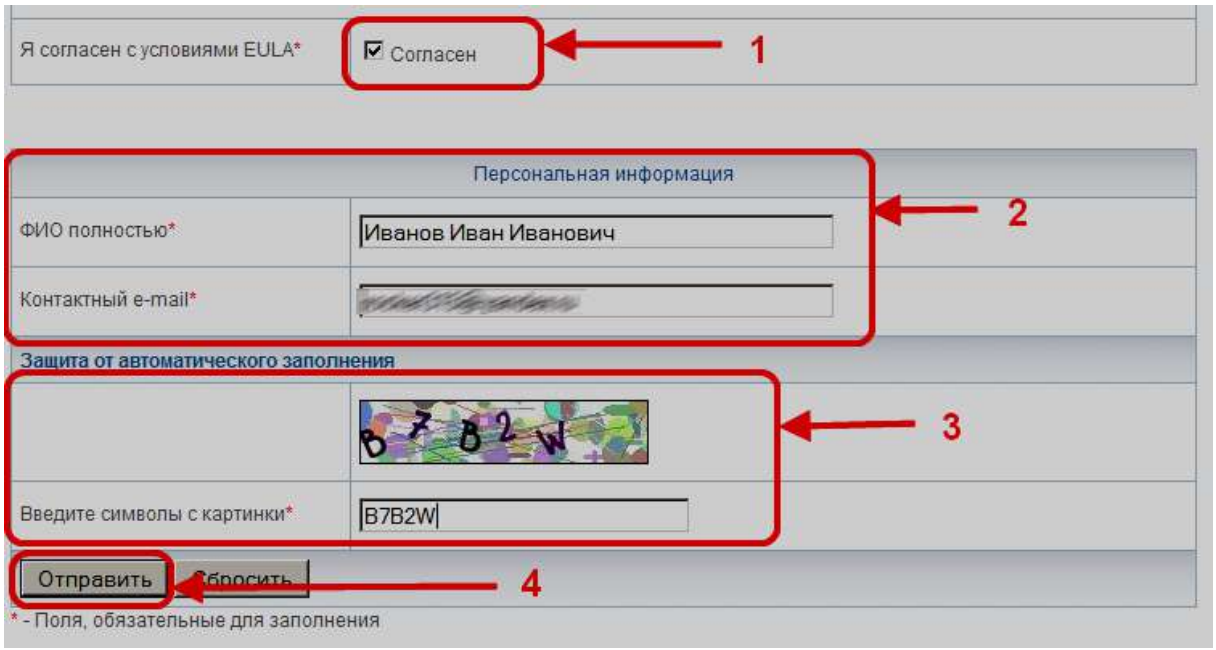


Рис. 11.36. **Процедура реєстрації VipNet CSP**

3. Перейдіть за отриманим посиланням (рис. 11.37, позиція 1) для скачування продукту і **збережіть** вказаний серійний номер (див. рис. 11.37, позиція 2). Скачайте продукт можна і з папки рекомендованого софтвера.

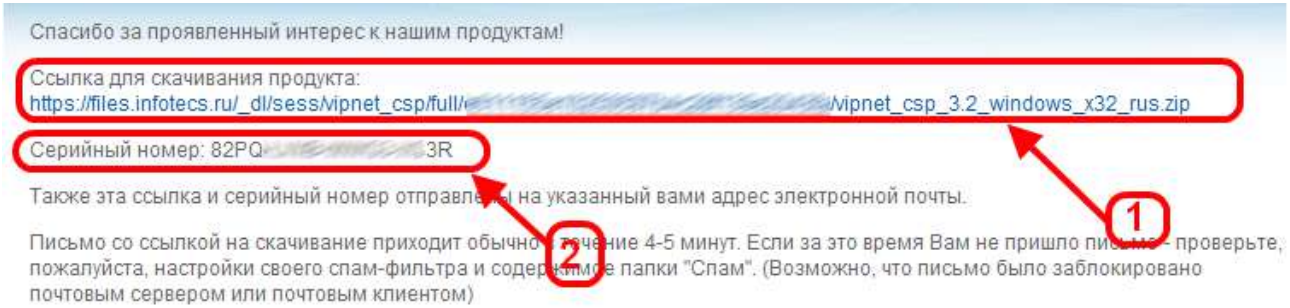


Рис. 11.37. **Скачування і збереження серійного номера ViPNet CSP**

4. Дочекайтеся закінчення скачування дистрибутиву і запустіть установлення ViPNet CSP файлом Setup.exe з папки архіву ViPNet_CSP_3.2_windows_x32_rus.zip (або ViPNet_CSP_3.2_windows_x64_rus.zip).

5. Виконайте встановлення ViPNet CSP, дотримуючись інструкцій майстра встановлення.

6. Після перезавантаження комп'ютера запустіть налаштування ViPNet CSP із панелі "Пуск".

7. Зареєструйте ViPNet CSP:

1) виберіть "Зареєструвати ViPNet CSP" і натисніть "Далі" (рис. 11.38);

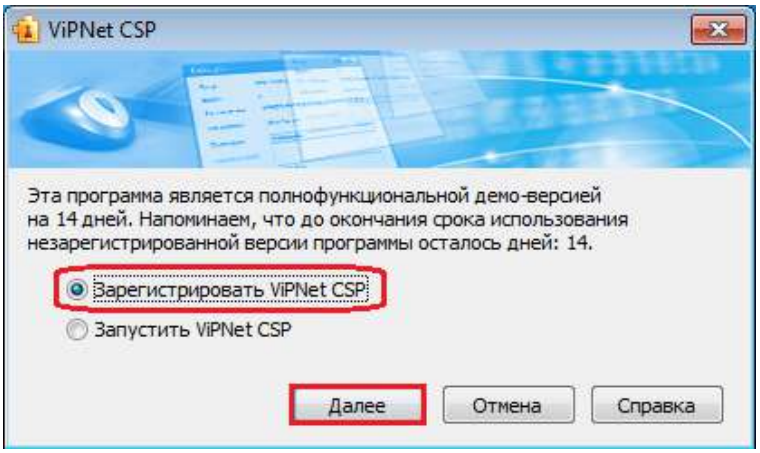


Рис. 11.38. **Процедура реєстрації ViPNet CSP (етап 1)**

2) виберіть "Запит на реєстрацію (отримайте код реєстрації)" і натисніть "Далі" (рис. 11.39);

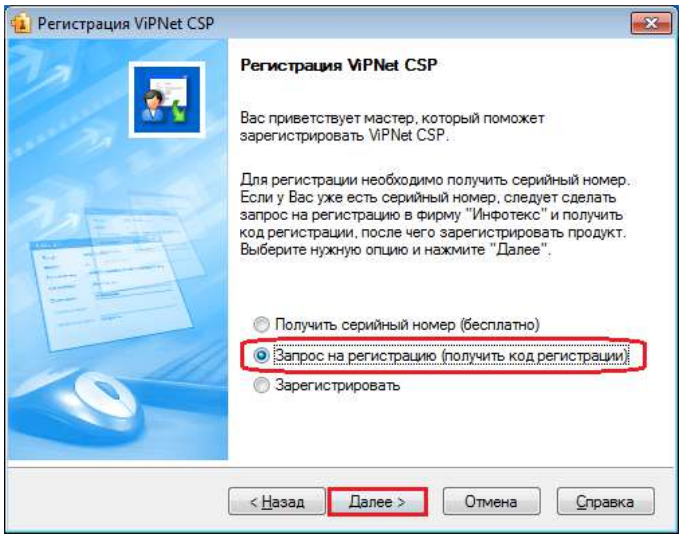


Рис. 11.39. **Процедура реєстрації VipNet CSP (етап 2)**

3) виберіть "Через Інтернет (online)" і натисніть "Далі" (рис. 11.40);

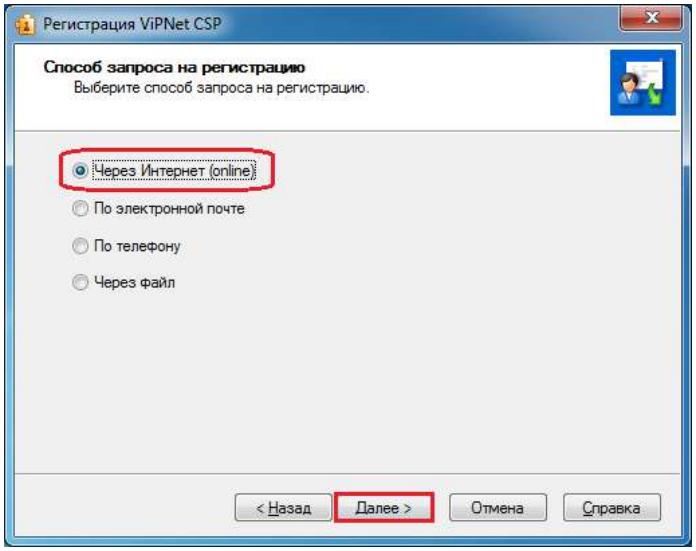


Рис. 11.40. **Процедура реєстрації VipNet CSP (етап 3)**

4) заповніть форму своїми реєстраційними даними, включаючи "Серійний номер" ViPNet CSP, отриманий під час реєстрації на етапі 3 (див. рис. 11.40) (так само серійний номер можна знайти на e-mail, указаному під час завантаження ViPNet CSP із web-сайта) і натисніть "Далі" (рис. 11.41);

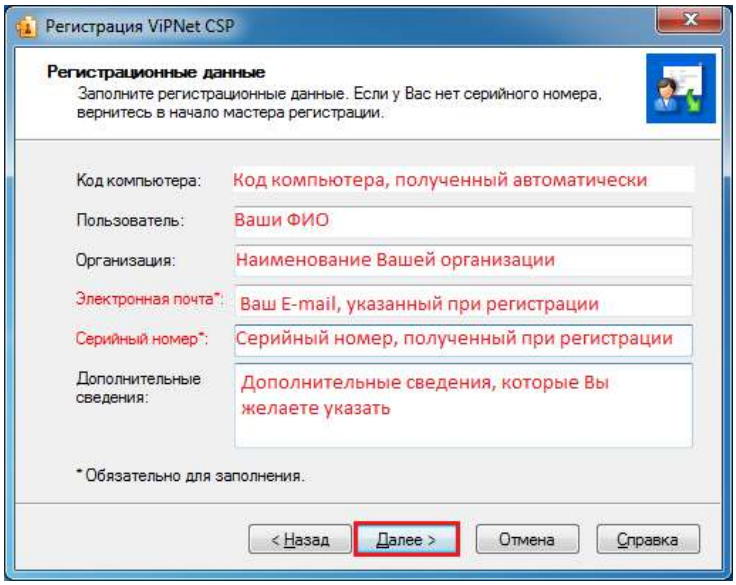


Рис. 11.41. **Процедура реєстрації даних VipNet CSP (етап 4)**

5) запустіть процес запиту на реєстрацію (рис. 11.42);

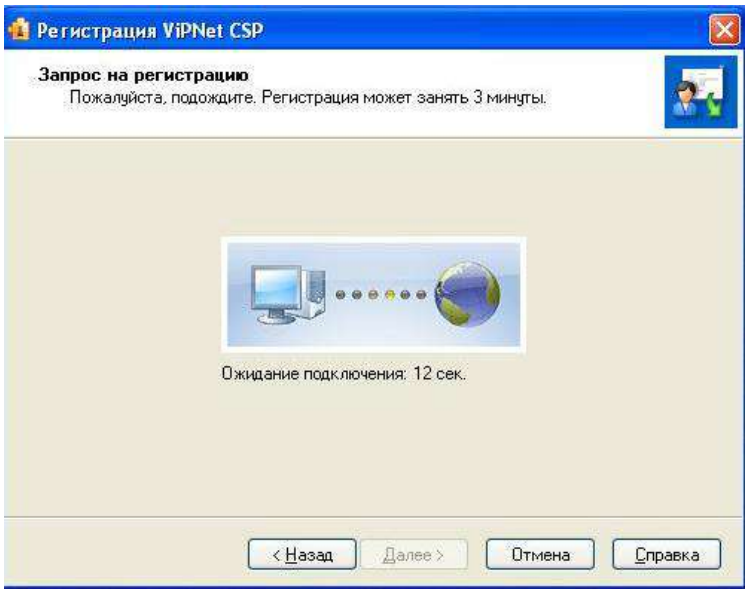


Рис. 11.42. **Процедура запиту реєстрації**

Увага! Якщо під час виконання дії вказаного на цьому етапі в разі натиснення на кнопку "Далі" (див. рис. 11.41) процес очікування виконання запиту займає більш ніж 3 хв і з'являється таке повідомлення (рис. 11.43):

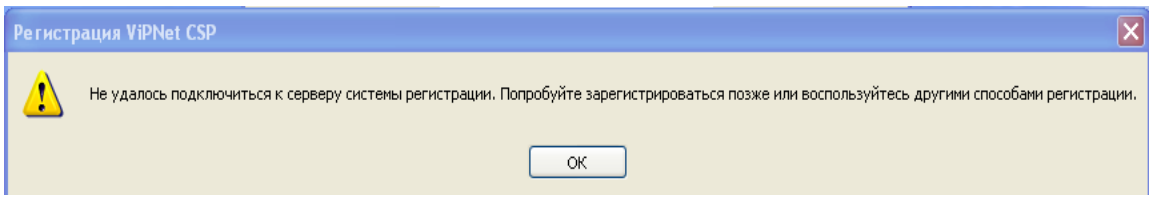


Рис. 11.43. **Помилка під час реєстрації VipNet CSP**

То в цьому разі необхідно:

а) переконатися в доступності мережі Інтернет, якщо вона не доступна, то необхідно усунути причину відмови, звернувшись до системного адміністратора або в компанію, що надає доступ до цієї мережі;

б) якщо Інтернет доступний, повернутися у вікно вибору способу реєстрації (див. рис. 11.41) і скористатися альтернативними способами реєстрації: "За електронною поштою" або "За телефоном", дотримуючись інструкцій майстра реєстрації.

Наново зробіть запит (online) (див. рис. 11.41 і 11.42).

6) у разі успішної реєстрації, натисніть "Готово" (рис. 11.44);

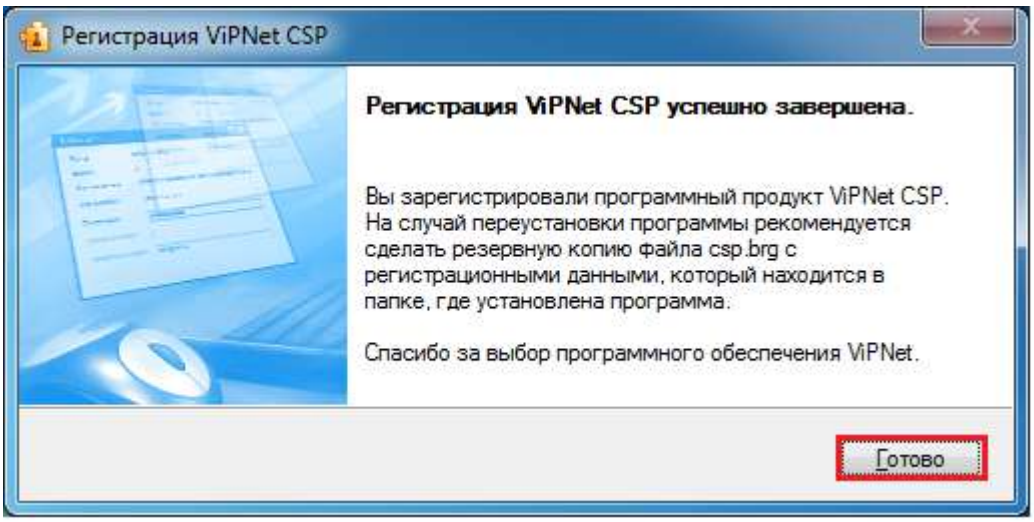


Рис. 11.44 **Успішна реєстрація VipNet CSP**

7) на запитання "Запустити ViPNet CSP зараз?" дайте відповідь Yes або запустіть ViPNet CSP пізніше з панелі "Пуск".

Налаштування VipNet CSP для роботи з електронним цифровим підписом

Установлення особистого сертифіката

Скопіюйте облікові дані (папка "Прізвище ім'я по батькові") із компакт-диска (або з іншого зовнішнього носія) у будь-яке місце жорсткого диска комп'ютера.

Із головного вікна ViPNet CSP перейдіть на закладку "Контейнери" (рис. 11.45).

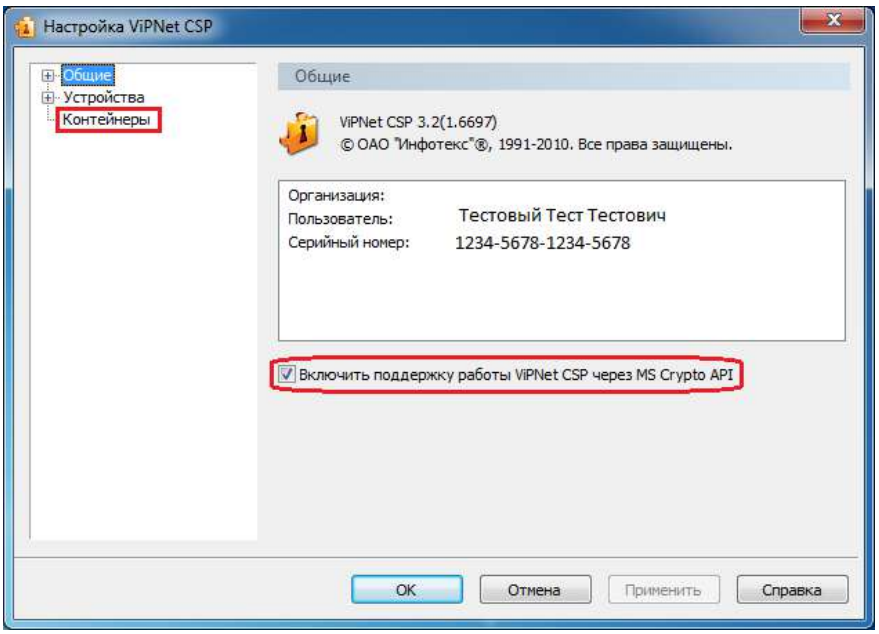


Рис. 11.45. **Головне вікно VipNet CSP**

Натисніть "Додати", укажіть місцеположення ключів (папка "Прізвище ім'я по батькові"), виберіть "Ім'я контейнера" виду "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" (єдиний файл у випадному списку без розширення) і натисніть OK (рис. 11.46 і 11.47).

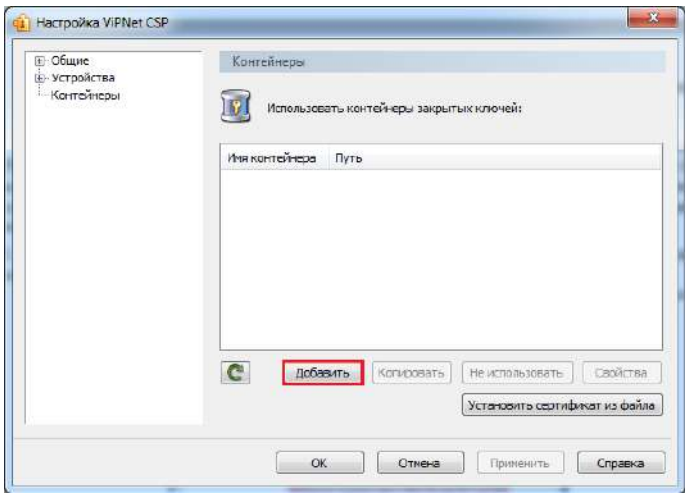


Рис. 11.46. **Вибір папки із ключем**

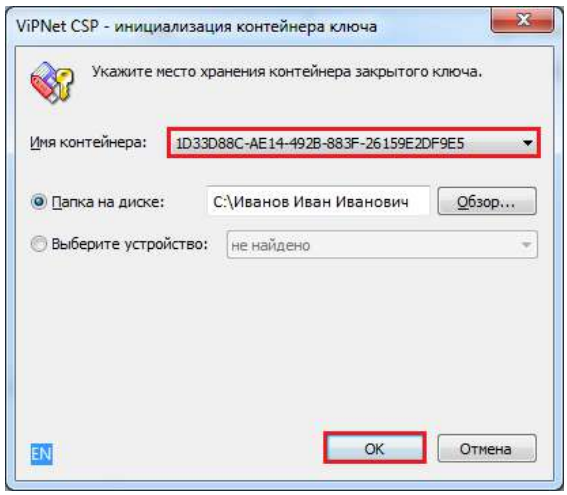


Рис. 11.47. **Вибір контейнера**

ViPNet CSP видасть повідомлення "Контейнер успішно додано" і поставить запитання про встановлення знайдених у контейнері сертифікатів у системне сховище. Натисніть "Так" і перейдіть до наступного пункту інструкції (рис. 11.48).

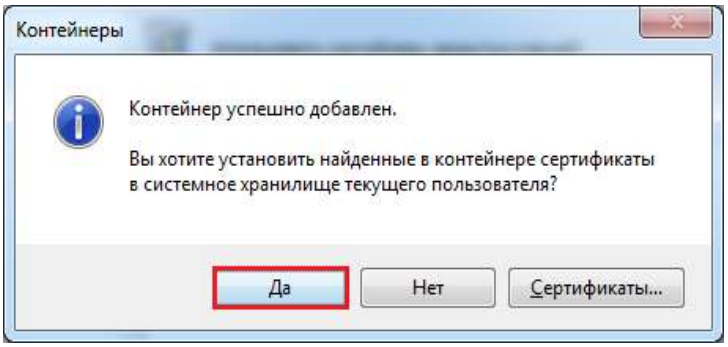


Рис. 11.48. **Діалогове вікно успішного додавання контейнеру**

Увага! Рекомендовано змінити пароль доступу до контейнера зі стандартного 123456 на більш стійкий, який будете знати тільки ви. Для цього на вкладці "Контейнери" виділіть ваш контейнер, натисніть "Властивості" → "Змінити пароль" (рис. 11.49 і 11.50) і дотримуйтеся інструкцій майстра.

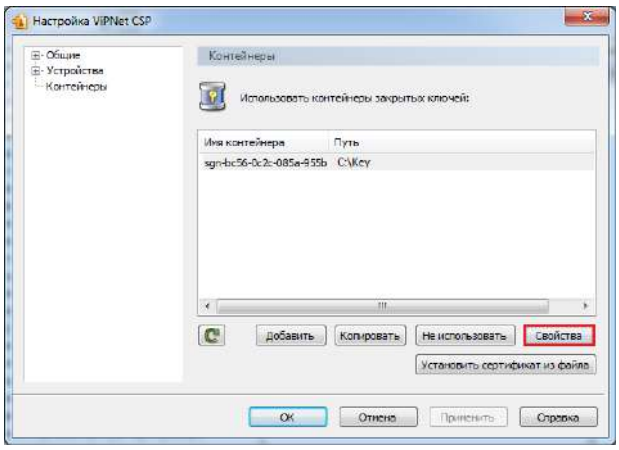


Рис. 11.49. **Вікно налаштування ViPNet CSP**

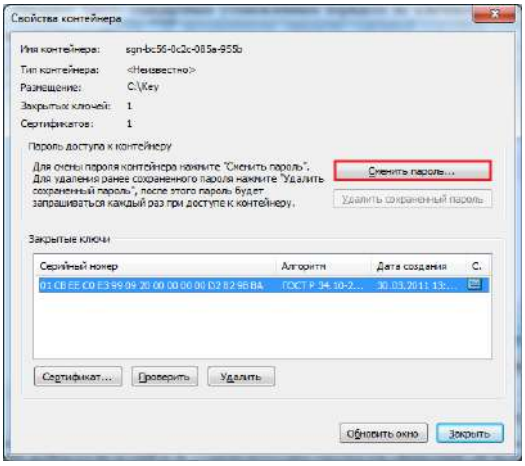


Рис. 11.50. **Вікно властивостей контейнера**

Установлення кореневого сертифіката і списків відгуку сертифікатів засвідчувального центру

1) в інтернет-оглядачі відкрийте посилання http://iitrust.ru/services/uc/sert.php (рис. 11.51);

2) знайдіть у таблиці рядок "УЦ ІЇТ (К2) (XXXX) (82)" (див. рис. 11.51, позиція 1);

3) скачайте кореневий сертифікат УЦ (див. рис. 11.51, позиція 2) і Списки відкликаних сертифікатів (див. рис. 11.51, позиція 3):

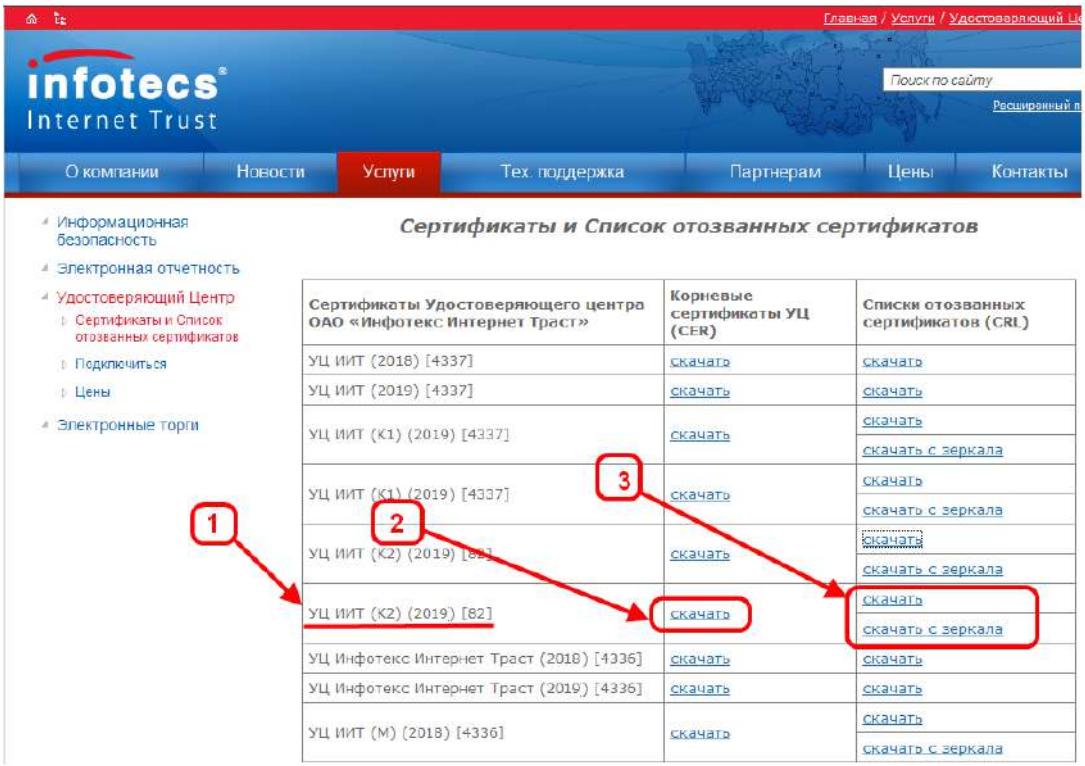


Рис. 11.51. **Посилання на скачування кореневого сертифіката**

Завантаження й установлення кореневого сертифіката

Під час відкриття посилання (див. рис. 11.51, позиція 2) відкрийте кореневий сертифікат безпосередньо з вікна web-браузера, натиснувши "Відкрити" (рис. 11.52).

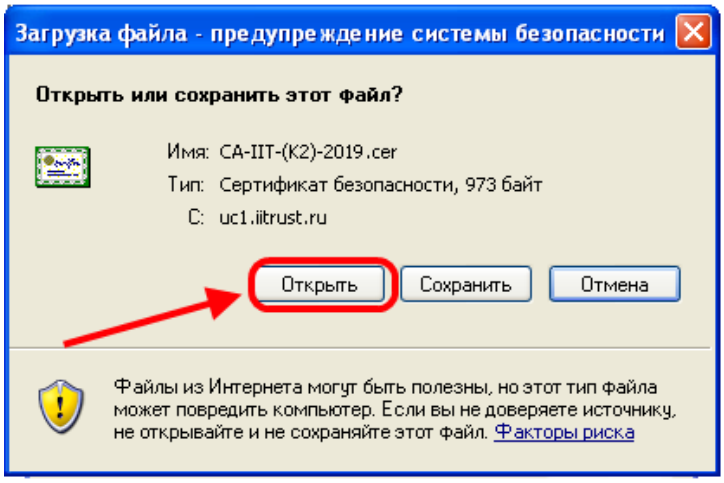


Рис. 11.52. **Завантаження кореневого сертифіката**

Натисніть "Установити" (рис. 11.53).

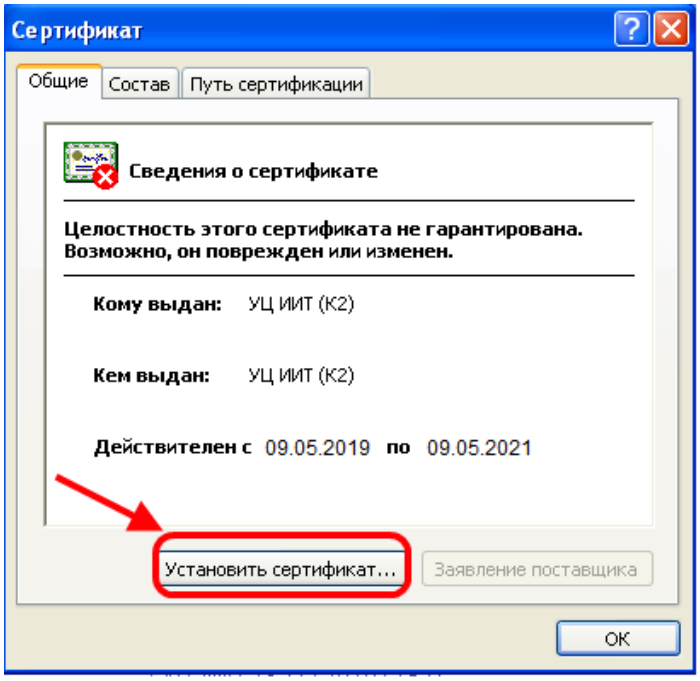


Рис. 11.53. **Установлення кореневого сертифіката**

Коли відкриється майстер імпорту сертифікатів Windows, натисніть "Далі" (рис. 11.54).

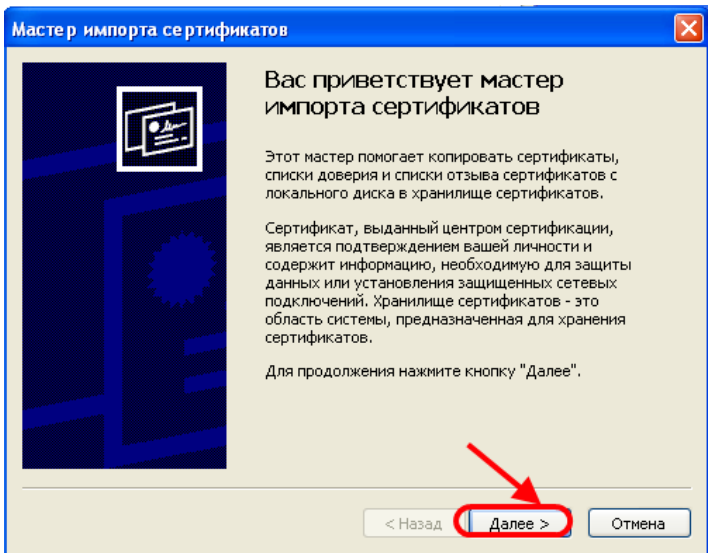


Рис. 11.54. **Майстер імпорту сертифіката**

Виберіть "Помістити всі сертифікати в таке сховище" (рис. 11.55, позиція 1). Натисніть "Огляд" (рис.11.55, позиція 2), виберіть "Довірені кореневі центри сертифікації" (рис. 11.55, позиція 3). Натисніть ОК (рис. 11.55, позиція 4).

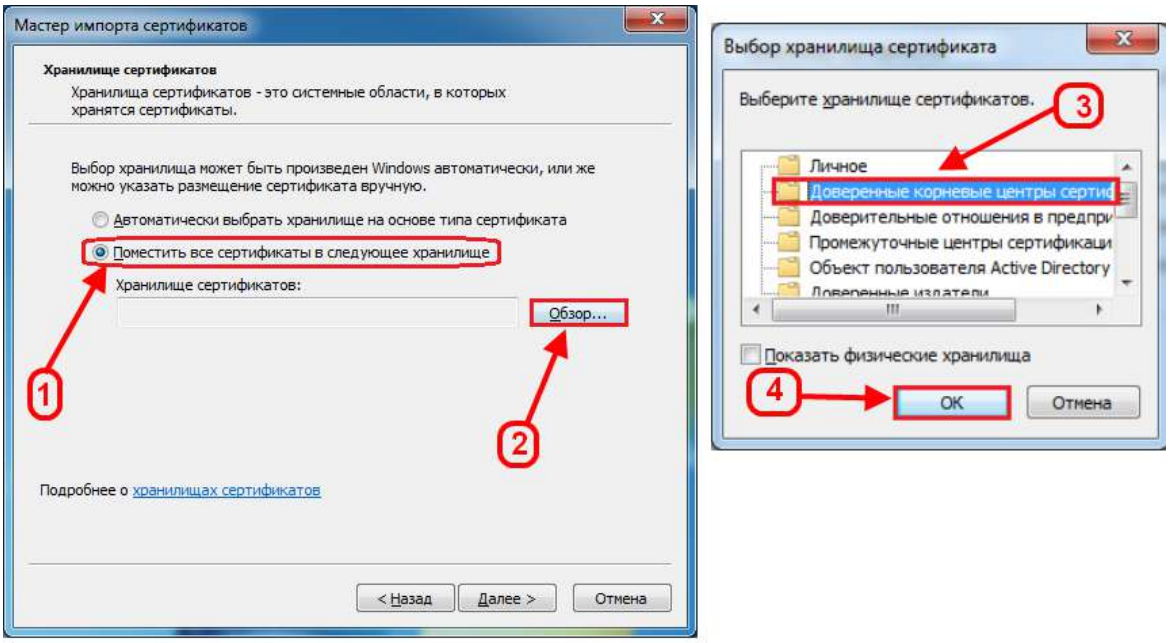


Рис. 11.55. **Приміщення всіх сертифікатів у сховищі**

У вікні попередження системи безпеки натисніть "Так" (рис. 11.56).

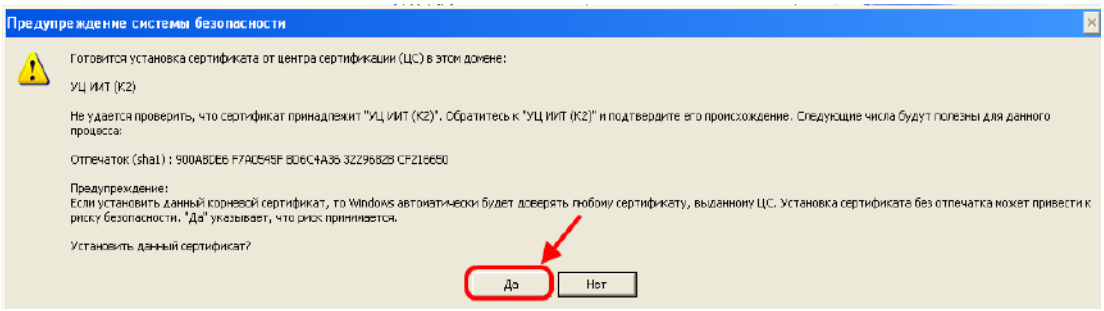


Рис. 11.56. **Попередження системи безпеки**

У разі успішного імпорту кореневого сертифіката з'явиться сповіщення, натиснути ОК (рис. 11.57).

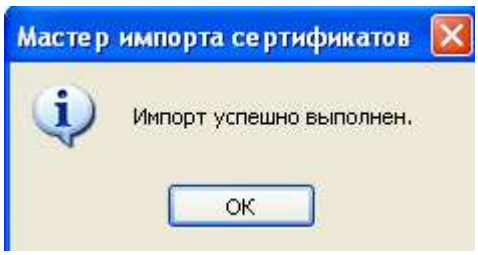


Рис. 11.57. **Успішний імпорт кореневого сертифіката**

Закрийте вікно кореневого сертифіката (рис. 11.58).

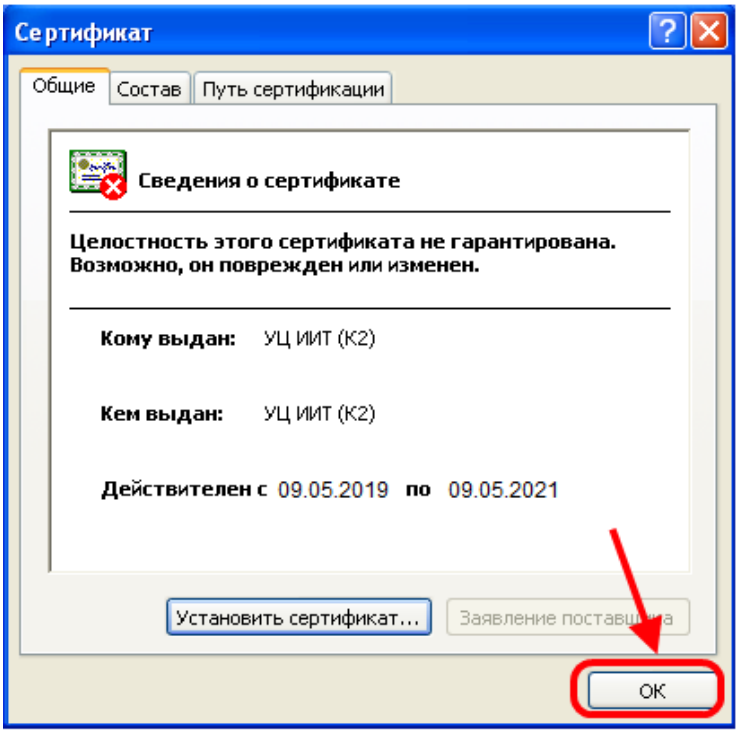


Рис. 11.58. **Вікно кореневого сертифіката**

Завантаження й установлення списку відгуку сертифікатів УЦ

Під час відкриття посилання (див. рис. 11.51, позиція 3), натисніть "Зберегти" (рис. 11.59).

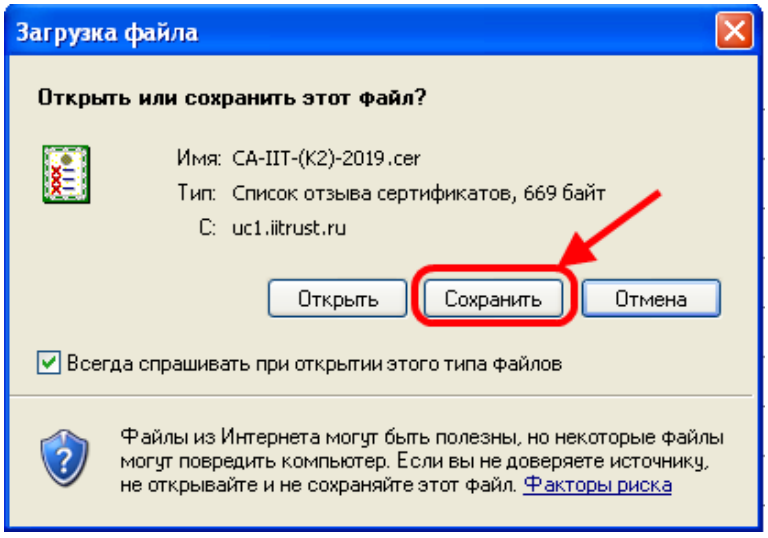


Рис. 11.59. **Установлення списку відгуку сертифікатів**

Збережіть список відгуку сертифікатів, наприклад на *Робочому столі* (рис. 11.60, позиції 1 і 2).

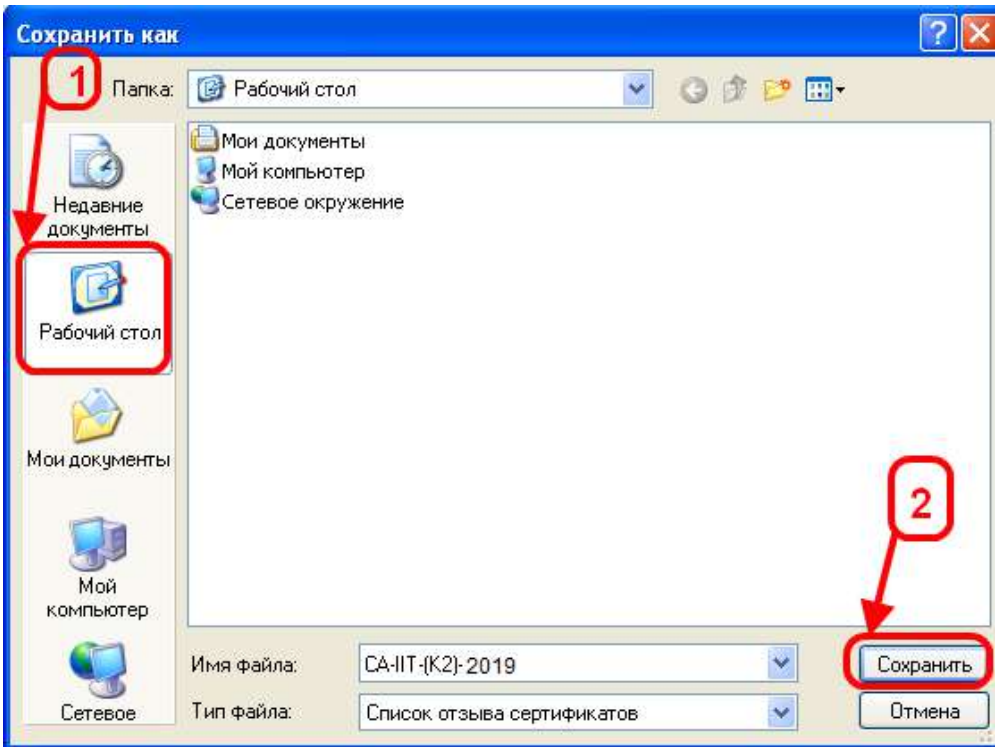


Рис. 11.60. **Збереження списку відгуку сертифікатів**

Закрийте вікно після завершення завантаження (рис. 11.61).

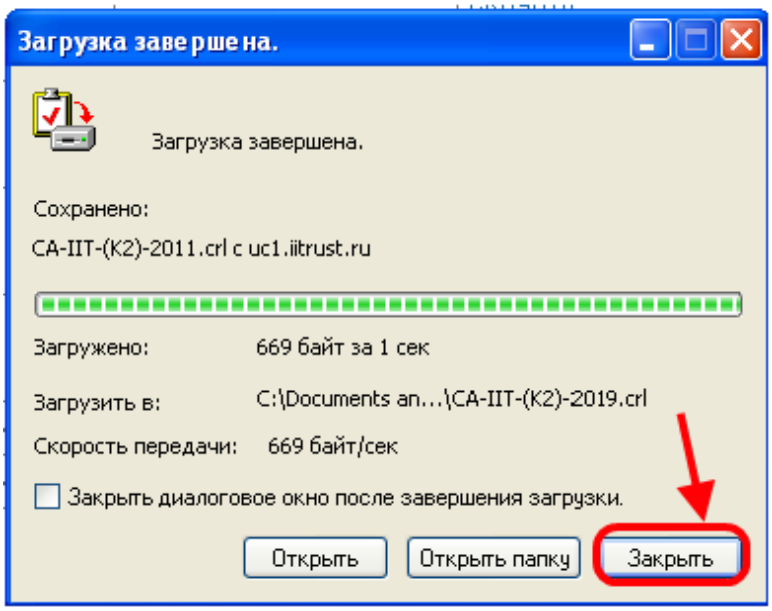


Рис. 11.61. **Завершення завантаження списку відгуку сертифікатів**

Виділіть завантажений і збережений на *Робочому столі* список відгуку сертифікатів правою кнопкою мишки (рис. 11.62, позиція 1) і з випадного меню виберіть "Установити список відгуку (CRL)" (рис. 11.62, позиція 2).

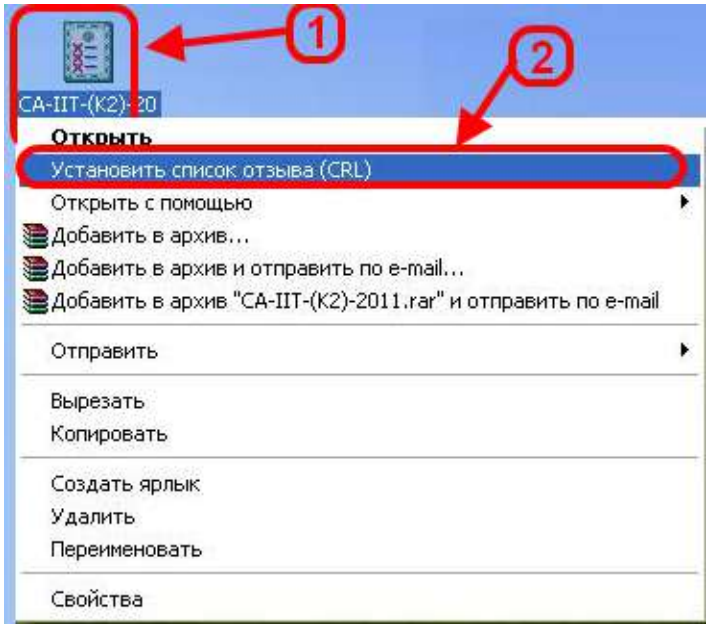


Рис. 11.62. **Установлення списку відгуку сертифікатів**

Коли відкриється майстер імпорту сертифікатів Windows, натисніть "Далі" (рис. 11.63).

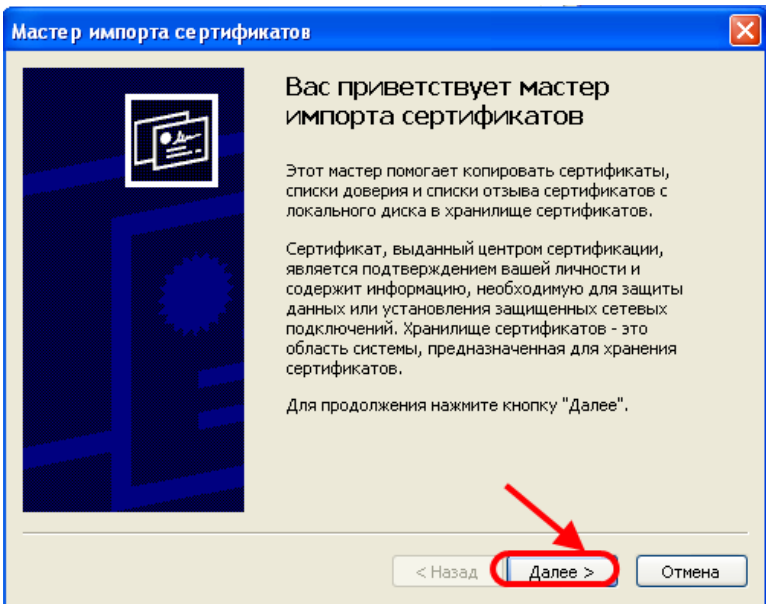


Рис. 11.63. **Майстер імпорту сертифікатів**

Виберіть "Помістити всі сертифікати в таке сховище" (рис. 11.64, позиція 1). Натисніть "Огляд" (рис. 11.64, позиція 2). Виберіть "Проміжні центри сертифікації" (рис.11.64, позиція 3). Натисніть Ок (рис. 11.64, позиція 4).

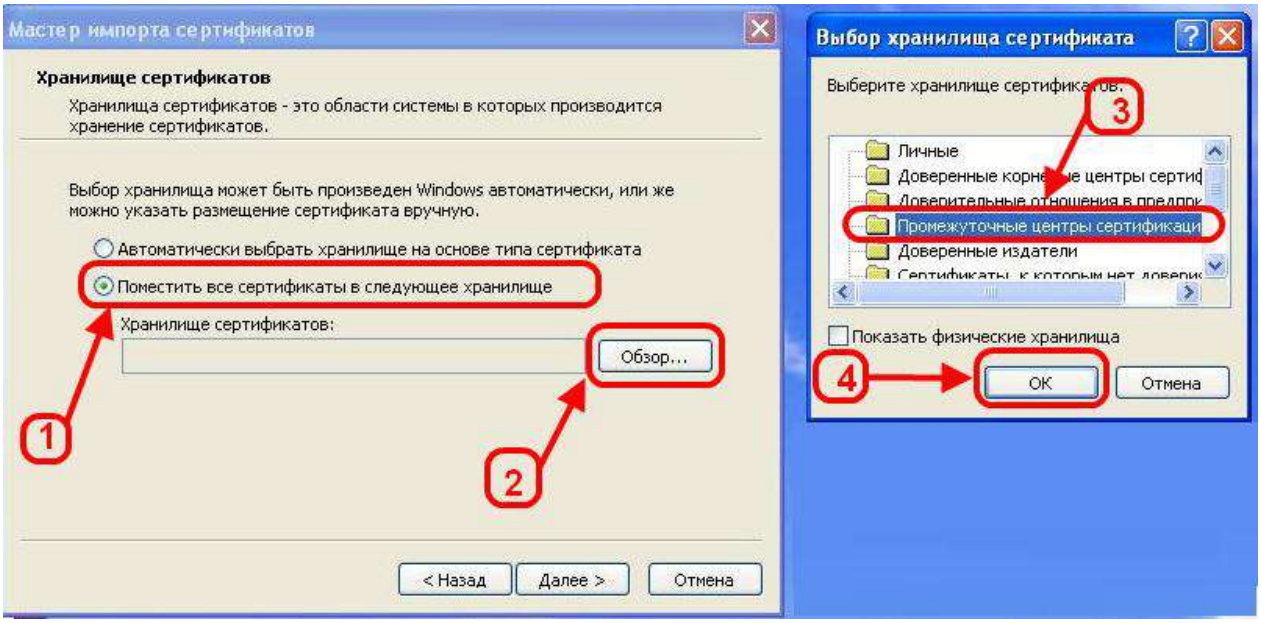


Рис. 11.64. **Приміщення сертифікатів у сховищі**

У разі успішного імпорту кореневого сертифіката з'явиться сповіщення, натиснути ОК (рис. 11.65).

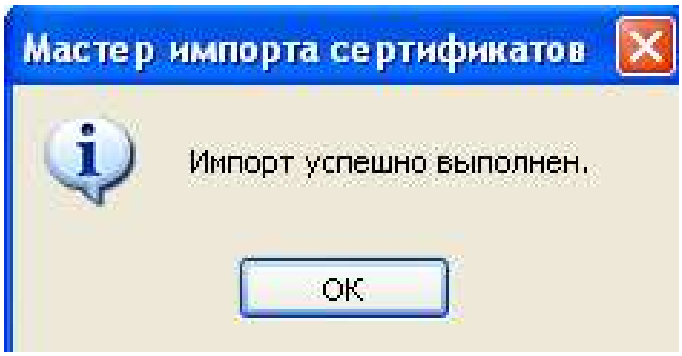


Рис. 11.65. **Успішне завершення встановлення списку відгуку сертифікатів**

На цьому процедуру встановлення кореневого сертифіката і списків відгуку сертифікатів засвідчувального центру завершено.

Загальне завдання для виконання

Перед початком виконання роботи встановити ВСЕ задане програмне забезпечення

1. Дослідна частина

Етап 1. Шифрування текстового файла

За допомогою блокнота підготуйте текстовий файл `Test.txt`, який має містити прізвище, ім'я і по батькові студента, назву факультету і номер групи. Наприклад, *Коваленко Микола Миколайович, економічної інформатики, група. 6.04.122.010.19.1*. Використовуючи заданий у п. 1 індивідуального завдання алгоритм шифрування і його програмну реалізацію з описаних у цьому керівництві, виконайте шифрування. Підрахуйте контрольні суми кожного файла, підтверджуючи його цілісність, і занесіть результати до табл. 11.2. Зробіть підтвердження результатів шифрування відповідними скриншотами.

Таблиця 11.2

Результати шифрування текстового файла

№ п/п	Ім'я файла	Алгоритм шифрування	Пароль (ключ)	Розмір файла, байт	Контрольна сума	Пояснення
1	`Test.txt`	–	–	78	ATSM-0MTH	Початковий файл
2						Зашифровані файли
3						
4						

Етап 2. Дешифрування текстового файла

Під час аналізу викраденого текстового файла, заданого в п. 2. індивідуального завдання, виявилося, що його зашифровано. У ході спілкування з його автором удалося з'ясувати тільки те, що під час шифрування використано простий пароль: *qwerty*. Заданих у роботі програмних засобів **досить** для відкриття текстового файла.

Зробіть підтвердження результатів дешифрування відповідними скриншотами. Обов'язково вкажіть назву програми, за допомогою якої вдалося дешифрувати, та алгоритм.

2. Аналітична частина

Етап 3. Злам парольного захисту файла

У результаті закачування з Інтернету архівного файла, заданого в п. 3 індивідуального завдання, було з'ясовано, що він закритий паролем. Необхідно зламати пароль, ураховуючи, що він схожий на PIN-коди карток VISA службовців.

Використовуючи додаткову інформацію, удалося встановіть, що для документа office (.pdf) користувач у вигляді пароля задав номер телефона. Визначте пароль.

Необхідно зламати документ і подати у звіті обов'язкове значення паролів, вивести вміст файлів та зробити підтвердження результатів зламу відповідними скриншотами.

Етап 4. Виконання крекінгу тестового виконуваного файла TestP.exe

Виконайте зміну базового тестового пароля, який прописано всередині файла TESTP.exe на значення kaixo (6B 61 69 78 6F), а також зробити так, щоб під час уведення правильного ключа видавалася повідомлення про помилку, а під час уведення будь-якого іншого (неправильного) ключа відбувалася правильна умовна реєстрація програми.

Супроводжуйте процес крекінгу відображенням скриншотів з обов'язковим поданням підсумкового виду вікон Breakpoints і Patches програми OllyDbg.

Етап 5. Отримання і реєстрація сертифіката електронного цифрового підпису

Виконайте установлення криптопровайдера центру сертифікації на свій комп'ютер. Отримайте 2 сертифікати:

1. Кореневий сертифікат засвідчувального центру "Інфотекс Інтернет Траст".

2. Створіть власний сертифікат у засвідчувальному центрі "Інфотекс Інтернет Траст".

У звіті подайте скриншоти отриманих сертифікатів.

Додаткове завдання для виконання

Створіть додаток, у якому реалізують за допомогою CryptoAPI індивідуальне завдання, визначене в п. 4.

Індивідуальні завдання для виконання

Варіант № 1

1. Cast (128), Triple DES (192), Serpent (128).
2. Enc-01.txt.
3. Варіант-01.rar.
4. Виконайте злам пароля, яким зашифровано файл, за заданих фіксованих параметрів (виберіть самостійно та відобразіть у додатку), наприклад, за довжини пароля, що дорівнює 8 символам, алгоритму гешування CALG_SHA тощо.

Варіант № 2

1. Triple DES (192), Serpent (128), Skipjack.
2. Enc-02.txt.
3. Варіант-02.zip.
4. Розробіть діалогове вікно задавання параметрів цифровому підпису.

Варіант № 3

1. Serpent (128), Skipjack, AES-256.
2. Enc-03.txt.
3. Варіант-03.rar.
4. Досліджуйте параметри криптопровайдеров, установлених на комп'ютері.

Варіант № 4

1. Vigenure, Enigma, RC4.
2. Enc-04.txt.
3. Варіант-04.zip.
4. Перелічіть усі криптопровайдери, установлені на комп'ютері.

Варіант № 5

1. Enigma, RC4, SEED.
2. Enc-05.txt.

3. Варіант-05.rar.

4. Побудуйте генератор ключів на основі введених, призначених для користувача даних.

Варіант № 6

1. Triple DES (168), IDEA, SHAKAL-2.

2. Enc-06.txt.

3. Варіант-06.zip.

4. Дослідіть параметри вибраного користувачем криптопровайдера зі встановленого на комп'ютері.

Варіант № 7

1. Skipjack, AES-256, RC2.

2. Enc-07.txt.

3. Варіант-07.rar.

4. Виконуйте обмін сесійними ключами з іншим екземпляром додатка за іменованим каналом і висвічення отриманого ключа.

Варіант № 8

1. AES-256, Substitution, Cobra128 .

2. Enc-08.txt.

3. Варіант-08.zip.

4. Побудуйте тестувальну процедуру, початковими даними якої був би цифровий підпис, а процедура перевіряла, відповідно до встановлених параметрів правильність цього підпису. За наявності помилки за потреби виправляйте її.

Варіант № 9

1. Substitution, Cobra128, DES.

2. Enc-09.txt.

3. Варіант-09.rar.

4. Дослідіть не менше ніж 4 алгоритми гешування і подайте результати порівняння у клієнтській області вікна.

Варіант № 10

1. DESX, Scytale, RC2.

2. Enc-10.txt.

3. Варіант-10.zip.

4. Розробіть процедуру верифікації (перевірки) сертифіката відкритого ключа (PKI), що міститься у файлі *.cer, і подання результатів такої перевірки.

Варіант № 11

1. MARS, Twofish (128), IDEA.

2. Enc-11.txt.

3. Варіант-11.rar.

4. Розробіть додаток, який би на основі даних із системного реєстру, виводив би у клієнтську область і файл на диску список усіх криптопровайдерів з описом використовуваних ними алгоритмів, довжин ключів (для геш-функцій – розмірів блоків), ідентифікаторів цих алгоритмів.

Варіант № 12

1. Cobra128, DES, Vigenure.

2. Enc-12.txt.

3. Варіант-12.zip.

4. Створіть цифровий підпис у форматі PKSC # 7 за допомогою функції CryptSignMessage() шляхом відкриття сховища сертифікатів (CertOpenStore()) та вибору потрібного сертифіката (CertFindCertificateInStore()).

Варіант № 13

1. RC4, SEED, Cast (256).

2. Enc-13.txt.

3. Варіант-13.rar.

4. Організуйте отримання ідентифікатора безпеки поточного користувача.

Варіант № 14

1. DES, Vigenure, Enigma.

2. Enc-14.txt.

3. Варіант-14.zip.

4. Розробіть програму, у якій ідентифікатор безпеки перетвориться на символьне (String) уявлення, використовуючи функцію ConvertSidToStringSid. Задайте графічний інтерфейс для програми.

Варіант № 15

1. SEED, Cast (256), DESX.

2. Enc-15.txt.

3. Варіант-15.rar.

4. Розробіть програму, у якій ініціалізувався ідентифікатор безпеки, використовуючи функцію initiaiizesid. Причому заздалегідь за допомогою функції GetSidLengthRequired визначте довжину ідентифікатора безпеки. Після ініціалізації ідентифікатора безпеки за допомогою функції GetSidSubAuthority визначте індекс відносного ідентифікатора, після чого встановіть його значення. Потім перевірте правильність ідентифікатора безпеки за допомогою виклику функції IsValidSid.

Варіант № 16

1. 3-DES, AES-128, Twofish (256).

2. Enc-16.txt.

3. Варіант-16.zip.

4. Дослідіть привілеї власного облікового запису користувача. Надайте можливість установлення не менше ніж 10 різних привілеїв.

Варіант № 17

1. Twofish (256), Twofish (128), RC6.

2. Enc-17.txt.

3. Варіант-17.rar.

4. Розробіть програму, яка виводить у графічному режимі вміст ідентифікатора безпеки.

Варіант № 18

1. Affine Shift, Modular Shift Algoritm, ARCFOUR.

2. Enc-18.txt.

3. Варіант-18.zip.

4. Розробіть додаток, який би на основі даних із системного реєстру виводив би у клієнтську область і файл на диску список усіх криптопровайдерів з описом використовуваних ними алгоритмів, довжин ключів (для геш-функцій – розмірів блоків), ідентифікаторів цих алгоритмів.

Варіант № 19

1. Serpent (256), Affine Shift, Modular Shift Algoritm.

2. Enc-19.txt.

3. Варіант-19.rar.

4. Розробіть програму, яка визначає ім'я облікового запису та ім'я домену за ідентифікатором безпеки облікового запису, використовуючи для цього функцію LookupAccountSid.

Варіант № 20

1. Twofish (128), Scytale, Blowfish (448).

2. Enc-20.txt.

3. Варіант-20.zip.

4. Розробіть програму, яка визначає ідентифікатор безпеки та ім'я домена за ім'ям облікового запису, використовуючи для цього функцію LookupAccountName.

Варіант № 21

1. Rail Fence Algoritm, RC6, Serpent (256).

2. Enc-21.txt.

3. Варіант-21.rar.

4. Розробіть програму, у якій ініціалізувався дескриптор безпеки для нового каталогу. Водночас власник каталогу і первинна група власника каталогу визначаються операційною системою, використовуючи механізм, заданий за замовчуванням. У цьому разі власником каталогу є користувач, від імені якого запущено програму. Оскільки списки управління доступом створюються порожніми, то доступ до каталогу дозволено всім користувачам.

Варіант № 22

1. RC5, Caesar, Camelia.

2. Enc-22.txt.

3. Варіант-22.zip.

4. Розробіть діалогове вікно задавання параметрів цифровому підпису.

Варіант № 23

1. IDEA, SHAKAL-2, AES-192.

2. Enc-23.txt.

3. Варіант-23.rar.

4. Дослідіть параметри криптопровайдеров, установлених на комп'ютері.

Варіант № 24

1. Caesar, Camelia, Cast (128).
2. Enc-24.txt.
3. Варіант-24.zip.
4. Перелічіть усі криптопровайдери, установлені на комп'ютері.

Варіант № 25

1. SHAKAL-2, AES-192, MARS.
2. Enc-25.txt.
3. Варіант-25.rar.
4. Побудуйте генератор ключів на основі введених, призначених для користувача даних.

Варіант № 26

1. Camelia, Cast (128), Triple DES (192).
2. Enc-26.txt.
3. Варіант-26.zip.
4. Дослідіть параметри вибраного користувачем криптопровайдера зі встановленого на комп'ютері.

Варіант № 27

1. RC6, Serpent (256), Affine Shift.
2. Enc-27.txt.
3. Варіант-27.rar.
4. Виконайте обмін сесійними ключами з іншим екземпляром додатка за іменованим каналом і висвічення отриманого ключа.

Варіант № 28

1. AES -128, Twofish (256), Rail Fence Algoritm.
2. Enc-28.txt.
3. Варіант-28.zip.
4. Побудуйте тестувальну процедуру, початковими даними якої був би цифровий підпис, а процедура перевіряла, відповідно до встановлених параметрів, правильність цього підпису. За наявності помилки за потреби виправляйте її.

Варіант № 29

1. ARCFOUR, Blowfish (448), RC5.
2. Enc-29.txt.

3. Варіант-29.rar.

4. Дослідіть не менше ніж 4 алгоритми гешування і подайте результати порівняння у клієнтській області вікна.

Варіант № 30

1. Blowfish (448), RC5, Caesar.

2. Enc-30.txt.

3. Варіант-30.zip.

4. Організуйте отримання ідентифікатора безпеки поточного користувача.

Контрольні запитання

1. Що таке "сесійний ключ"?

2. Поясніть порядок отримання сесійного ключа.

3. У чому специфіка підпису документів?

4. Із якою метою використовують цифровий електронний підпис?

5. Чим відрізняється закритий ключ від сесійного ключа?

6. Наведіть приклади криптопровайдерів.

7. Опишіть процес створення криптопровайдера.

8. Назвіть порядок обчислення контрольної суми файла.

9. Яким чином взаємодіють між собою алгоритм шифрування та сесійний ключ?

10. Дайте визначення поняттю "блоб".

11. Назвіть можливі типи ключів для шифрування.

12. Який криптопровайдер в операційній системі Windows використовують за замовчуванням?

13. Поясніть основні математичні операції, які використовують в алгоритмі шифрування AES?

14. Поясніть основні математичні операції, які використовують в алгоритмі шифрування RC4?

15. Поясніть основні математичні операції, які використовують в алгоритмі шифрування DES?

16. Поясніть основні математичні операції, які використовують в алгоритмі шифрування Blowfish?

17. Поясніть основні математичні операції, які використовують в алгоритмі шифрування Cast?

18. Поясніть основні математичні операції, які використовують в алгоритмі шифрування Enigma?

19. Поясніть основні математичні операції, які використовують в алгоритмі шифрування SHA-1?

20. Поясніть основні математичні операції, які використовують в алгоритмі шифрування Skipjack?

21. Поясніть основні математичні операції, які використовують в алгоритмі шифрування IDEA?

22. Поясніть основні математичні операції, які використовують в алгоритмі шифрування Cobra128?

23. Поясніть основні математичні операції, які використовують в алгоритмі шифрування Twofish?

Лабораторна робота 12
Дослідження та оптимізація завантаження ОС Windows

Мета роботи: ознайомлення з етапами завантаження і завершення роботи ОС Windows на основі BIOS, особливостями та порядком переходу на технологію завантаження UEFI, аналізом аварійних дампів під час перебоїв і помилок роботи ОС.

Рекомендації з підготовки до виконання лабораторної роботи

Необхідно розглянути структури головного завантажувального запису ОС Windows, порядок проведення тестування пам'яті перед завантаженням, особливості побудови файлових систем FAT32, NTFS, ctc3 тощо. Рекомендується встановити досліджувальну операційну систему Windows на віртуальну машину, використовуючи додатки VirtualBox, VirtualPC або ін.

Теоретичні відомості

Ще з часів Windows NT завантажувачем операційної системи був **NTLDR**, а починаючи з Windows Vista, його замінено новим диспетчером завантаження **BOOTMGR**. Викликано це тим, що NTLDR вже не годився для виконання завантаження системи на комп'ютерах, які використовують специфікацію Extensible Firmware Interface (EFI), покликану замінити базову систему введення-виведення BIOS.

Процес завантаження будь-якої операційної системи починається завжди однаково: після перевірки обладнання, управління отримує підпрограма BIOS (Basic Input/Output System), що прочитує із пристрою завантаження перший сектор, головним завантажувальним записом якого є **MBR** (**Master Boot Record).** Перед записом завантажувального сектора програма Setup має визначити формат розділу, оскільки від цього буде залежати вміст цього сектора. Для розділу, який відформатовано під NTFS, Windows записує код NTFScapable. Цей код надає Windows інформацію про структуру і формат диску та здійснює читання з файла BOOTMGR у кореневому каталозі тому. Після того як файл BOOTMGR опиниться в пам'яті, код NTFScapable передає управління його точці входу.

Диспетчер завантаження BOOTMGR є файлом невеликого розміру, розташованим у кореневому каталозі активного розділу. Основне його призначення – забезпечення подальшої процедури завантаження, відповідно до наявної **конфігурації**, що зберігають у спеціальному **сховищі даних конфігурації** (**BCD – Boot Configuratin Data)**, що є файлом з ім'ям **BCD** та перебувають у каталозі **BOOT** активного розділу.

Таким чином, для того щоб виконалося завантаження Windows із диспетчером BOOTMGR, активний розділ, як мінімум, має містити правильний завантажувальний запис **PBR** (**Partition Boot Record)**, файл диспетчера **BOOTMGR** і конфігураційні дані у файлі **\\BOOT\BCD**, системне **сховище конфігурації завантаження** (**BCD Store)**. У разі із завантаженням Windows, диспетчер **bootmgr** прочитує зі сховища конфігурації дані, необхідні для завантаження ядра системи, і передає управління додатку, що виконує наступний етап (**winload.exe**).

За своєю структурою, файл **\BOOT\BCD** є кущем реєстру, що відображають у редакторові реєстру Windows як розділ **HKLM\BCD000000** (рис. 12.1).

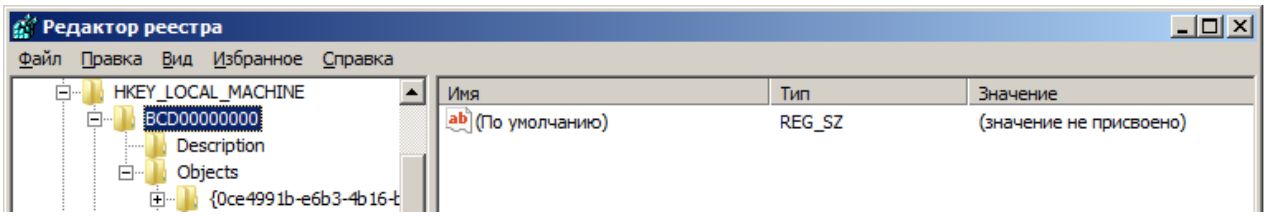


Рис. 12.1. **Відображення BCD Store в реєстрі**

Розділ конфігурації BCD містить підрозділ **Description** із параметрами опису і підрозділ **Objects** з об'єктами конфігурації завантаження.

Ці конфігурації завантаження можна умовно розподілити на 3 основні складові частини:

сховище BCD (Store);

записи у сховищі (Entries);

параметри записів (Entry Options).

Ієрархічно, сховищем конфігурації завантаження є сукупність об'єктів (Objects), що складаються з окремих елементів (Elements) (рис. 12.2).

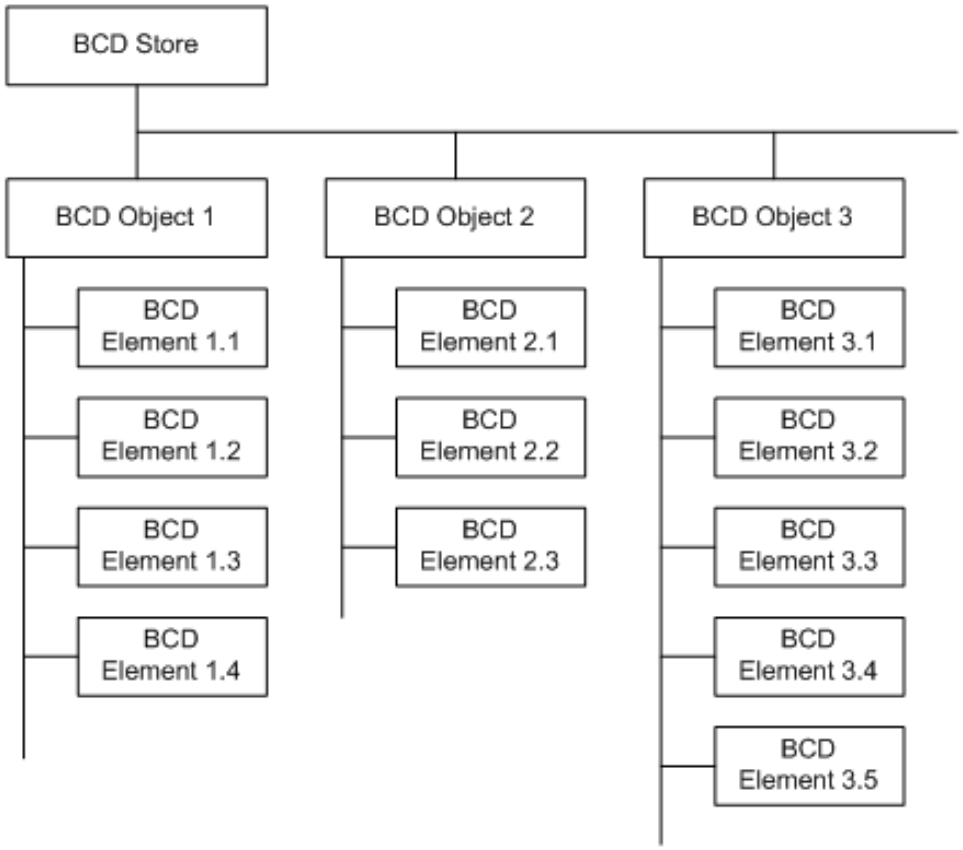


Рис. 12.2. **Ієрархічна структура BCD Store**

Кожен з об'єктів є впорядкованою структурою елементів, що обробляється диспетчером завантаження. Кожен об'єкт BCD має ім'я, котре є **глобальним унікальним ідентифікатором – GUID**. Ідентифікатор GUID формується програмним шляхом і однозначно є унікальним для тієї системи, де він створюється. Позначено GUID у вигляді груп із шістнадцяткових цифр, що розподіляють дефісами, і розміщених у фігурних дужках. Є 3 типи об'єктів:

додатки (application objects);

успадковані об'єкти (inheritable objects);

пристрої (device objects).

Об'єкти стандартних додатків конфігурації завантаження мають **зумовлені ідентифікатори**, що пов'язують деякі з ідентифікаторів GUID із внутрішніми ідентифікаторами (псевдонімами) редактора **bcdedit** (табл. 12.1).

Таблиця 12.1

Взаємозв'язок між GUID та іменами сховища

GUID	Ім'я BCDEDIT	Опис
{0ce4991b-e6b3-4b16-b23c-5e0d9250e5d9}	{emssettings}	
{1afa9c49-16ab-4a5c-4a90-212802da9460}	{resumeloadersettings}	
{1cae1eb7-a0df-4d4d-9851-4860e34ef535}	{default}	Default boot entry
{4636856e-540f-4170-a130-a84776f4c654}	{dbgsettings}	
{466f5a88-0af2-4f76-9038-095b170dc21c}	{legacy} {ntldr}	Legacy Windows Loader
{5189b25c-5558-4bf2-bca4-289b11bd29e2}	{badmemory}	
{6efb52bf-1766-41db-a6b3-0ee5eff72bd7}	{bootloadersettings}	
{7ea2e1ac-2e61-4728-aaa3-896d9d0a9f0e}	{globalsettings}	
{7ff607e0-4395-11db-b0de-0800200c9a66}	{hypervisorsettings}	
{9dea862c-5cdd-4e70-acc1-f32b344d4795}	{bootmgr}	Windows Boot Manager
{a5a30fa2-3d06-4e9f-b5f4-a01df9d1fcba}	{fwbootmgr}	Firmware Boot Manager
{ae5534e0-a924-466c-b836-758539a3ee3a}	{ramdiskoptions}	
{b2721d73-1db4-4c62-bf78-c548a880142d}	{memdiag}	Windows Memory Tester
{fa926493-6f1c-4193-a414-58f0b2456d1e}	{current}	Current boot entry

Кожен із розділів підрозділу **Objects** також складається із двох підрозділів: **Descriptions** з описом типу об'єкта й **Elements**, що визначає набір елементів із параметрами об'єкта. У розділі **Description** є ключ **Type** типу **REG_DWORD**, значення розрядів 28 – 31 якого визначає тип об'єкта (значення у старшій тетраді старшого байта):

1******* – додаток (табл. 12.2);

2******* – успадкований об'єкт (табл. 12.3);

3******* – пристрій (табл. 12.4).

Можна сказати з упевненістю, що команда **bcdedit /create** передбачає тільки такі комбінації (див. табл. 12.2).

Таблиця 12.2

Об'єкти додатка

Type	Параметри створення BCDEDIT
10100001	/create {fwbootmgr}
10100001	/create {fwbootmgr}
10100002	/create {bootmgr}
10200003	/create /application osloader
10200004	/create /application resume
10200005	/create {memdiag}
10300006	/create {legacy}/create {ntldr}
10400008	/create /application bootsector
10400009	/create /application startup

Таблиця 12.3

Успадковані об'єкти

Type	Параметри створення BCDEDIT
20100000	/create /inherit /create {badmemory} /create {dbgsettings} /create {emssettings} /create {globalsettings}
20200001	/create /inherit fwbootmgr
20200002	/create /inherit bootmgr
20200003	/create /inherit osloader /create {bootloadersettings} /create {hypervisorsettings}
20200004	/create /inherit resume /create {resumeloadersettings}
20200005	/create /inherit memdiag
20200006	/create /inherit ntldr
20200007	/create /inherit setupldr
20200008	/create /inherit bootsector
20200009	/create /inherit startup
20300000	/create /inherit device

Таблиця 12.4

Об'єкти пристроїв

Type	Параметри створення BCDEDIT
30000000	/create /device /create {ramdiskoptions}

Кожен BCD-елемент має свій власний ключ реєстру. Ім'я ключа подано у вигляді восьми шістнадцяткових цифр у форматі DWORD.

Імена розділів реєстру, пов'язаних з елементами об'єкта, типи даних і значення параметрів залежать від конкретної конфігурації завантаження, створеного для використання диспетчером **Bootmgr**. Якщо уважно придивитися до імен ключів, то можна помітити, що ім'я ключа пов'язано з його вмістом, так наприклад підрозділ з ім'ям **12000004** завжди містить рядковий параметр із текстовим описом елемента:

```
[HKEY_LOCAL_MACHINE\BCD00000000\Objects\{b2721d73-1db4-4c62-bf78-c548a880142d}\Elements\12000004]
"Element"="Діагностика пам'яті"
```

У табл. 12.5 наведено BCD-елементи, відомі в декількох сенсах. Так, у першому стовпці показано ключ (елемент) у вигляді числової константи, яка подає його в шістнадцятковому коді. Другий стовпець містить легке для читання символічне ім'я редактора **bcdedit.**

Таблиця 12.5

BCD-елементи

Константа	Ім'я елемента	Константа	Ім'я елемента	Константа	Ім'я елемента
1	2	1	2	1	2
11000001	device	14000006	inherit	15000012	debugaddress
12000002	path	14000008	recoverysequence	15000013	debugport
12000004	description	15000007	truncatememory	15000014	baudrate
12000005	locale	1500000B	integrityservices	15000015	channel
12000016	targetname	1500000C	firstmegabytepolicy	15000018	debugstart
12000019	busparams	1500000D	relocatephysical	15000022	emsport
12000030	loadoptions	1500000E	avoidlowmemory	15000023	emsbaudrate
1200004A	fontpath	15000011	debugtype	15000042	keyringaddress

1	2	1	2	1	2
15000047	configaccesspolicy	25000007	bootux	26000010	detecthal
15000051	initialconsoleinput	25000020	nx	26000020	displaybootmenu
15000052	graphicsresolution	25000021	pae	26000021	noerrordisplay
16000009	recoveryenabled	25000031	removememory	26000022	winpe
1600000B	badmemoryaccess	25000032	increaseuserva	26000024	nocrashautoreboot
1600000F	traditionalksegmappings	25000033	perfmem	26000025	lastknowngood
16000010	bootdebug	25000050	clustermodeaddressing	26000026	oslnointegritychecks
16000017	noumex	25000052	restrictapiccluster	26000027	osltestsigning
16000020	bootems	25000061	numproc	26000030	nolowmem
16000040	advancedoptions	25000063	configflags	26000040	vga
16000041	optionsedit	25000066	groupsize	26000041	quietboot
16000046	graphicsmodedisabled	25000071	msi	26000042	novesa
16000048	nointegritychecks	25000072	pciexpress	26000051	usephysicaldestination
16000049	testsigning	25000080	safeboot	26000054	uselegacyapicmode
16000050	extendedinput	250000C1	driverloadfailurepolicy	26000060	onecpu
1700000A	badmemorylist	250000E0	bootstatuspolicy	26000062	maxproc
21000001	osdevice	250000F0	hypervisorlaunchtype	26000064	maxgroup
21000001	filedevice	250000F3	hypervisordebugtype	26000065	groupaware
21000005	associatedosdevice	250000F4	hypervisordebugport	26000070	usefirmwarepcisettings
21000022	bcddevice	250000F5	hypervisorbaudrate	26000081	safebootalternateshell
22000001	bpbstring	250000F6	hypervisorchannel	26000090	bootlog
22000002	systemroot	25000100	tpmbootentropy	26000091	sos
22000002	filepath	25000120	xsavepolicy	260000A0	debug
22000002	applicationname	25000121	xsaveaddfeature0	260000A1	halbreakpoint
22000011	kernel	25000122	xsaveaddfeature1	260000A2	useplatformclock
22000012	hal	25000123	xsaveaddfeature2	260000B0	ems
22000013	dbgtransport	25000124	xsaveaddfeature3	260000F2	hypervisordebug
22000023	bcdfilepath	25000125	xsaveaddfeature4	260000F8	hypervisordisableslat
22000053	evstore	25000126	xsaveaddfeature5	27000030	customactions
220000F1	hypervisorpath	25000127	xsaveaddfeature6	31000003	ramdisksdidevice
23000003	default	25000128	xsaveaddfeature7	32000004	ramdisksdipath
23000003	resumeobject	25000129	xsaveremovefeature	35000001	ramdiskimageoffset
23000006	resumeobject	2500012A	xsaveprocessorsmask	35000002	ramdisktftpclientport
24000001	displayorder	2500012B	xsavedisable	35000005	ramdiskimagelength
24000002	bootsequence	26000001	pxesoftreboot	35000007	ramdisktftpblocksize
24000010	toolsdisplayorder	26000003	customsettings	35000008	ramdisktftpwindowsize
25000001	passcount	26000004	stampdisks	36000006	exportascd
25000002	testmix	26000004	pae	36000009	ramdiskmcenabled
25000003	failurecount	26000005	resume	3600000A	ramdiskmctftpfallback
25000004	timeout	26000005	cacheenable		
25000004	testtofail	26000006	debugoptionenabled		

Доступ до елемента можна здійснювати через інтерфейс інструментарію управління Windows (WMI). Для перегляду вмісту сховища конфігурації можна скористатися такою командою:

`bcdedit /enum all` – відобразити всі записи у BCD;

`bcdedit /enum all > C:\enum-all.txt` – те саме, що й у попередньому випадку, але з виведенням результатів у текстовий файл enum-all.txt на диску C.

Системний розділ – це розділ диска, на якому зберігають файли, необхідні для запуску Windows (наприклад, Ntldr, Boot.ini та Ntdetect.com) (рис. 12.3).

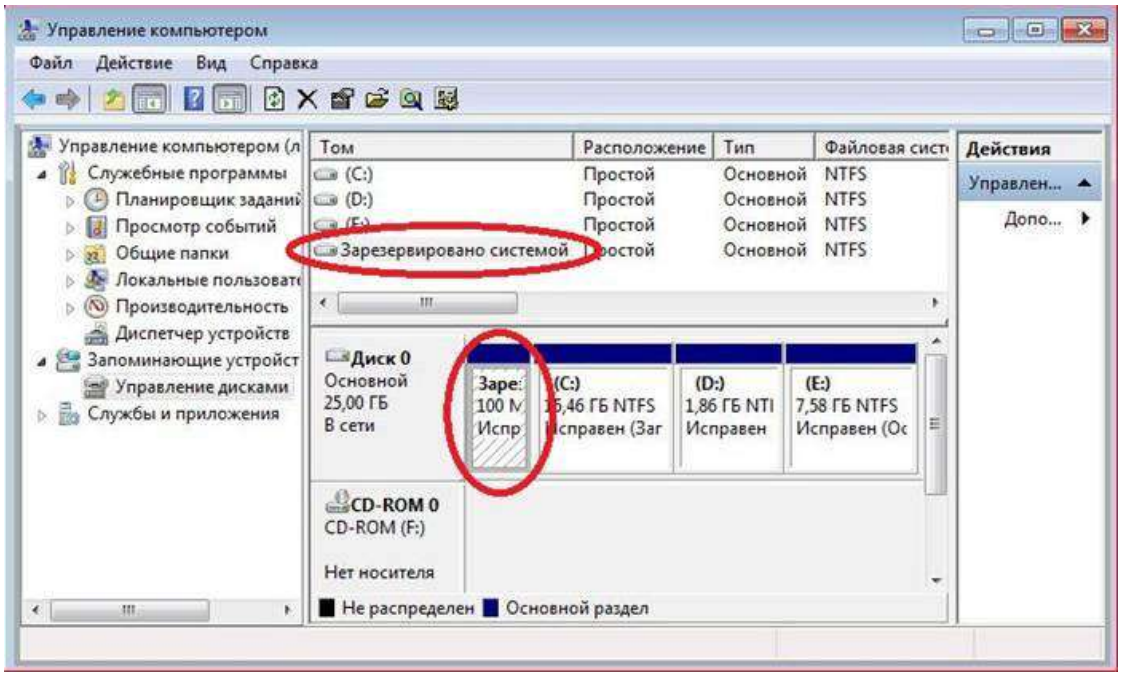


Рис. 12.3. **Структура фізичного диска в оснащенні Управління комп'ютером**

Обов'язкових умов для створення прихованого системного розділу Windows декілька:

1. Перша й основна умова – завантаження із зовнішнього пристрою (DVD, USB), оскільки під час запуску програми встановлення з-під Windows не буде можливості працювати з розділами жорсткого диска.

2. Загальна кількість основних (первинних) розділів жорсткого диска перед початком установлення не має перевищувати трьох. Тобто, якщо до початку встановлення простір жорсткого диска вже розподілено на 4 таких розділи, у цьому разі приховано розділ розміром 100 МБ сформовано не буде, а файли завантаження будуть перебувати на вже наявному

активному розділі. Причому це може бути не той розділ, на який установлено систему. Кількість логічних розділів на розширеному значення не має.

3. Розділ, у який виконують установлення, має бути першим за рахунком (верхнім, на графічному зображенні дискового простору).

4. Установлення операційної системи мають виконувати в нерозмічену область диска. Якщо диск уже розмічено, то під час вибору розділу для встановлення системи необхідно не просто його відформатувати, але й створити наново, тобто розділ для встановлення потрібно спочатку видалити. Відповідно, якщо не хочете, щоб на жорсткому диску створювався розділ **System Reserved**, не видаляйте той, що є.

До програм сторонніх виробників, розгляду яких присвячено лабораторну роботу, належать:

1. Victoria.
2. BOOTICE (by *www.ipauly.com*).
3. Windows Performance Toolkit.
4. Notmyfault.
5. Windows Debugger.

Victoria. Універсальна програма для тестування, діагностики та сервісного обслуговування IDE і Serial ATA вінчестерів. Орієнтована на широкий круг користувачів комп'ютерів. Працює з накопичувачем на низькому рівні (через порти контролера) (рис. 12.4).

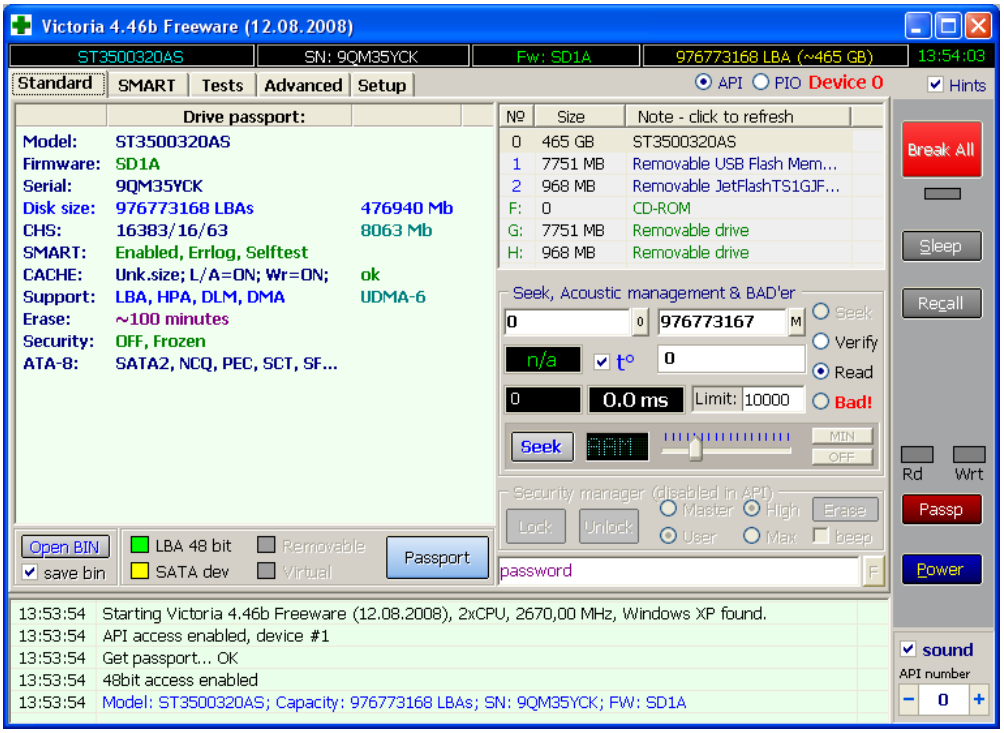


Рис. 12.4. **Головне вікно програми Victoria**

Основні можливості:

автодетект PCI ATA/SATA контролерів за кодом класу і підтримка 60 популярних моделей;

виведення повної технічної інформації про жорсткий диск;

10 тестів для перевірки поверхні й "механіки" диска;

створення і запис образу диска;

перевірка пам'яті й інтерфейсу HDD;

бенчмарк-функції;

дефектоскоп поверхні;

низькорівневе форматування HDD;

виявлення та приховування дефектів методом перепризначення секторів із резерву (remap);

посекторне копіювання довільної області HDD у файл із пропуском дефектних ділянок (може бути корисно для порятунку інформації з пошкодженого диска);

управління акустичним шумом;

управління парольним захистом;

зручний SMART-монітор;

можливість зміни об'єму HDD;

переглядання інформації про логічні розділи через порти;

убудований файловий диспетчер;

убудована довідкова система.

BOOTICE. Програма дозволить змінювати, створювати резервні копії та відновлювати пошкоджений головний завантажувальний запис Master Boot Record (MBR) і розділ Boot Record для локальних дисків або USB флеш-дисків (рис. 12.5).

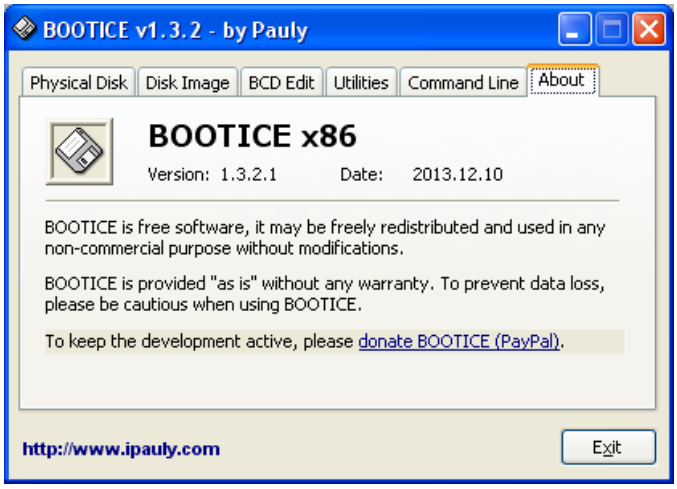


Рис. 12.5. **Головне вікно програми BOOTICE**

Також BOOTICE допоможе в розмічуванні та форматуванні USB жорстких дисків і флеш-карт, якщо раніше вони вже відформатували з файловою системою, яку не розпізнає операційна система Windows, що зазвичай приводить до того, що диск стає не видним у системі або видно не всі розділи. BOOTICE підтримує завантажувальні записи Grub4Dos, SysLinux, Plop, Windows NT5/6 та ін.

За допомогою функції Sector Editor в BOOTICE стає можливим відкриття редактора довільного сектора (рис. 12.6).

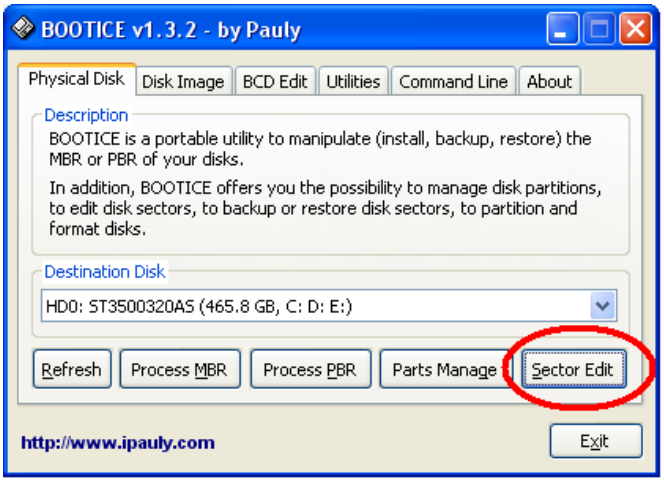


Рис. 12.6. **Функція Sector Editor у програмі BOOTICE**

Так, наприклад, MBR для фізичного диска 0 (HD0: ST3500320AS) буде мати такий вигляд (рис. 12.7):

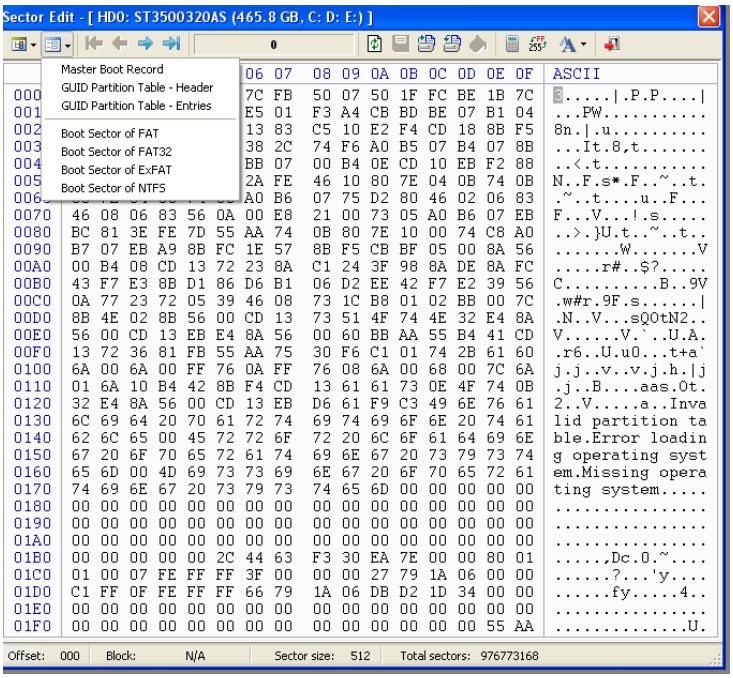


Рис. 12.7. **Уміст сектора Master Boot Record**

Здійснити аналіз розділів MBR жорсткого диска дозволить відповідне меню. У загальному випадку структура головного завантажувального запису MBR буде мати такий вигляд (рис. 12.8):

Програма і дані початкового завантажувача (код завантажувача)	Таблиця розділів диска	55AA

Рис. 12.8. **Структура таблиці головної завантажувальної записи**

Слід урахувати, що в разі, якщо користувач не знає що робити й до чого це приведе, то слід використовувати BOOTICE тільки зі змінними, несистемними жорсткими дисками.

У загальному вигляді структуру головного завантажувального запису MBR може бути подано таким чином (табл. 12.6):

програмний код і дані початкового завантажувача. (446 байтів);

таблиця розділів диска (4 поля по 16 байтів – 64 байти);

сигнатура 55AA (2 байти).

Таблиця 12.6

Структура головного завантажувального запису MBR

Зсув	Довжина, байтів	Опис	
0000h	446	Код завантажувача	
01BEh	16	Розділ 1	Таблиця розділів
01CEh	16	Розділ 2	
01DEh	16	Розділ 3	
01EEh	16	Розділ 4	
01FEh	2	Сигнатура (55h AAh)	

Завдання завантажувача проаналізувати таблицю розділів жорсткого диска, потім або передати управління завантажувальному коду активного розділу, або завантажити в RAM ядро операційної системи та передати йому управління. У таблиці розділів зберігається інформація про тип розділу і його розташування на жорсткому диску. Останні два байти MBR називають **сигнатурою**. Значення цих байтів має бути 55h AAh. У разі, якщо це не так, то запис прочитався некоректним. Структура опису розділу має такий вигляд (табл. 12.7):

Таблиця 12.7

Структура опису розділу MBR

Зсув	Довжина	Опис
00h	1	Ознака активності розділу
01h	1	Початок розділу – головка (Head)
02h	1	Початок розділу – сектор (біти 0 – 5), циліндр (біти 6 і 7)
03h	1	Початок розділу – циліндр (старші біти 8 і 9 зберігаються в байті номера сектора)
04h	1	Код типу розділу (Partition type indicator)
05h	1	Кінець розділу – головка
06h	1	Кінець розділу – сектор (біти 0 – 5), циліндр (біти 6 і 7)
07h	1	Кінець розділу – циліндр (старші біти 8 і 9 зберігаються в байті номера сектора)
08h	4	Зсув першого сектора (Preceding sectors)
0Ch	4	Кількість секторів розділу (Section in partition)

Ознака активності – це ознака, що позначає можливість завантаження операційної системи із цього розділу. Для стандартних завантажувачів може набувати таких значень (рис. 12.9):

80h – розділ є активним (active partition);

00h – розділ є неактивним;

інші значення є помилковими, їх ігнорують.

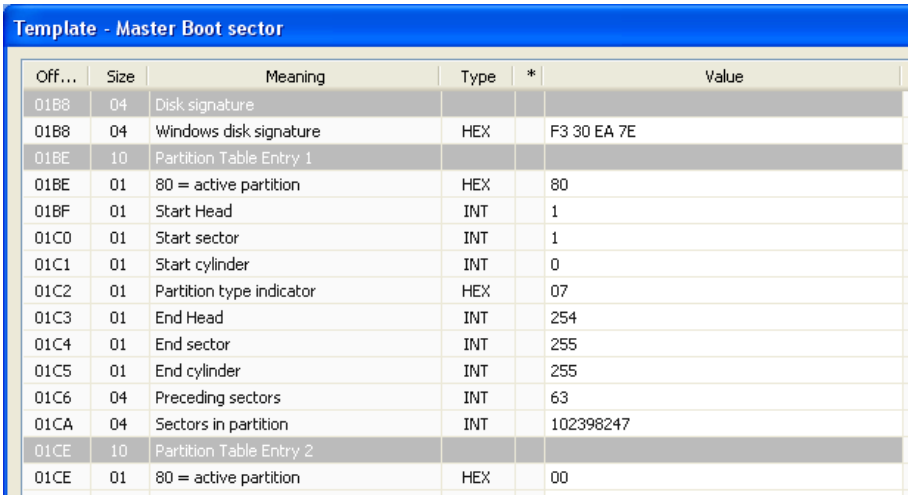


Рис. 12.9. **Структура MBR у програмі BOOTICE**

Координати початку і кінця розділу подано у CHS-форматі (циліндр, головка, сектор). CHS не дозволяє виконувати адресацію більш ніж до 7,8 ГБ даних, для адресації до розділів, що перебувають за межами 7,8 ГБ, використовують LBA-адресацію.

Код типу розділу визначає код файлової системи, використовуваної в цьому розділі (табл. 12.8).

Таблиця 12.8

Коди типу розділу

Коди	Типи розділів
00h	Порожній запис (вільне місце)
01h	FAT – 12
02h	XENIX root
03h	XENIX usr
04h	FAT– 16 до 32 МБ
05h	Розширений розділ
06h	FAT – 16 понад 32 МБ
07h	Windows NT NTFS (і деякі інші – тип визначають за вмістом завантажувального запису)
08h	AIX
09h	AIX завантажувальний
0Ah	OS/2 Boot диспетчер
0Bh	FAT– 32
0Ch	FAT– 32 з використанням LBA
0Eh	FAT– 16 із використанням LBA (VFAT)
0Fh	Розширений розділ LBA (те саме що й 05h із використанням LBA)

Windows Performance Toolkit – це програмне забезпечення, яке дозволяє значно прискорити роботу персонального комп'ютера. Windows за допомогою цієї утиліти працює швидше та стабільніше. Можливе відновлення втрачених даних і повний контроль над жорсткими дисками. Згладжує перебої роботи системи й усуває її недоліки. Стирає історію відвідин сайтів, а так само очищає жорсткий диск від непотрібних файлів.

Windows Performance Toolkit поширюють у складі Windows SDK – після встановлення SDK із каталогу BIN необхідно встановити версію Windows Performance Toolkit, відповідну для платформи, – wpt_x86.msi, wpt_x64.msi або wpt_ia64.msi, відповідно.

В основі Windows Performance Toolkit лежать утиліти:

XPerf, яка слугує для активації збирання інформації;

XBootMgr, що дозволяє збирати інформацію у процесі завантаження комп'ютера, його вимкнення, переходу в режим Standby та виходу із цього режиму;

XPerfView, використовувана для візуального аналізу інформації про продуктивність.

Взаємодію XPerf і XPerfView, насамперед, із підсистемою **Event Tracing for Windows** (ETW), показано на рис. 12.10.

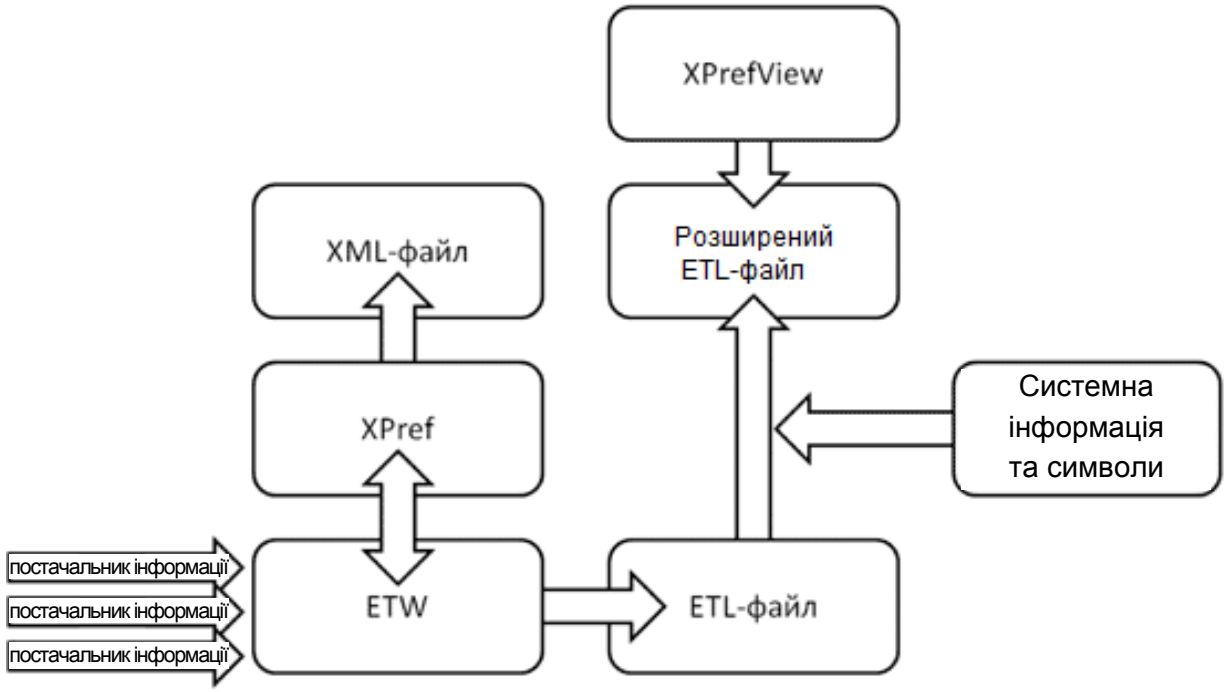


Рис. 12.10. **Взаємодія утиліт XPerf і XPerfView із системними компонентами**

У загальному випадку робота з утилітами XPerf і XPerfView відбувається таким чином:

1. За допомогою утиліти XPerf включають протоколювання подій на рівні ETW.
2. Виконують операції, що цікавлять нас, процеси тощо.

3. За допомогою утиліти XPerf завершують сесію протоколювання.

4. Виконують оброблення інформації – додавання до ETL-файла, що створюється підсистемою ETW, системної інформації та символів.

5. Переглядають і аналізують результат за допомогою утиліти XPerfView.

Аналогічним чином працює й утиліта XBootMgr. Дані, що збираються цією утилітою, допоможуть, наприклад, визначити процеси, що збільшують час переходу комп'ютера з одного стану в інший, роботи, що впливають на коректне завершення тощо.

Наприклад, якщо ETL-файл буде розміщуватися в каталозі C:\Trace. Тоді процес збирання даних про **завантаження системи** запускається однією командою:

```
xbootmgr -trace boot -traceFlags BASE+CSWITCH+DRIVERS
+POWER -resultPathC:\Trace.
```

Аналогічні команди можна використовувати для діагностики **гібернації**:

```
xbootmgr -trace hibernate -traceFlags BASE+CSWITCH+DRI
VERS+POWER -resultPathC:\Trace.
```

і **сну**:

```
xbootmgr -trace standby -traceFlags BASE+CSWITCH+DRI
VERS+POWER -resultPathC:\Trace.
```

Комп'ютер буде перезавантажено. Якщо після входу до системи буде запит UAC від xbootmgr, то необхідно дозволити утиліті продовжити роботу. Через дві хвилини відобразиться приблизно таке вікно (рис. 12.11).

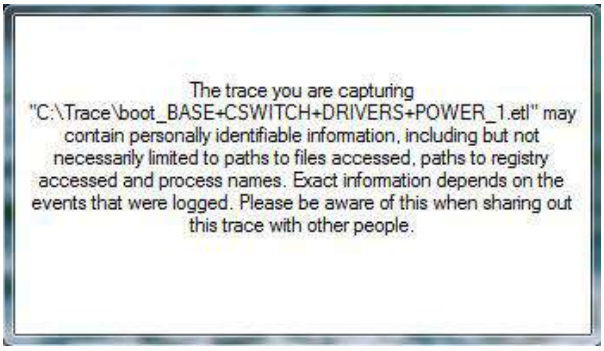


Рис. 12.11. **Вікно попередження про порядок застосування XPerf**

Якщо воно закриється, у каталозі C:\Trace мають бути три файли, як показано на рис. 12.12.

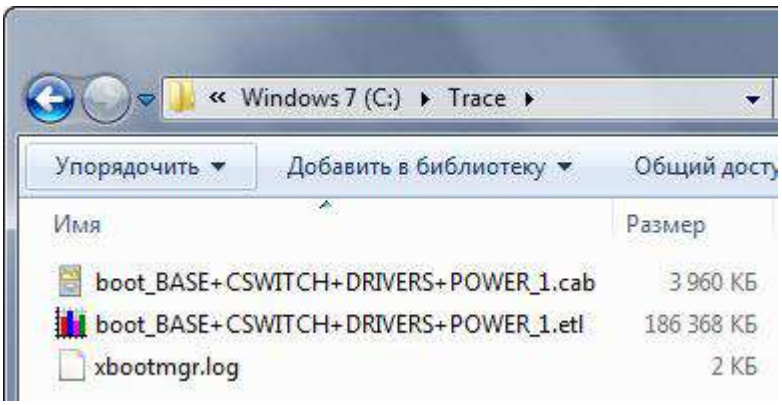


Рис. 12.14. **назва**

Рис. 12.12. **Підготовлені файли для аналізу XPerfView**

Для аналізу ETL-файла використовують Performance Analyzer (утиліту **XPerfView).**

Графік **Boot Phases** відображає тривалість основних етапів завантаження. На ньому видно, що останній етап, **Post Boot** зайняв 24,7 с (Duration), а загальний час завантаження становив майже 80 с (End Time) (рис. 12.13):

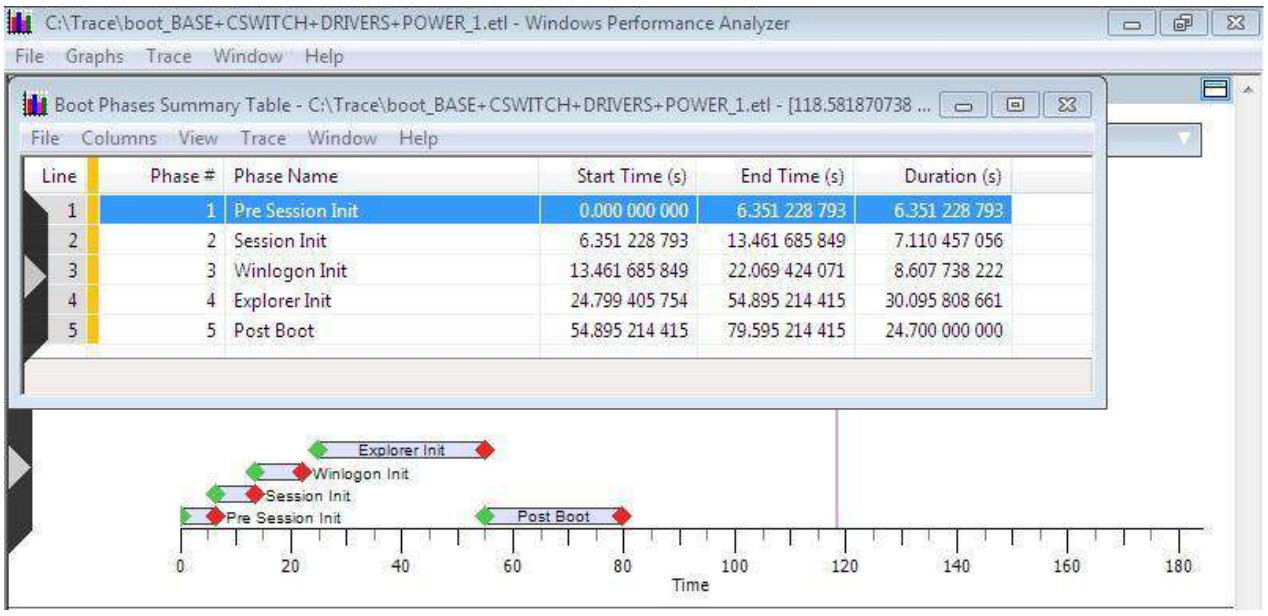


Рис. 12.13. **Аналіз фаз завантаження за графіком Boot Phases**

На рис. 12.14 показано основні етапи завантаження, причому головний із них складається із чотирьох фаз.

Етап **OSLoader** слідує відразу після ініціалізації BIOS. Візуально він починається після заставки та діагностичних екранів BIOS, а закінчується приблизно з появою екрана "Завантаження Windows".

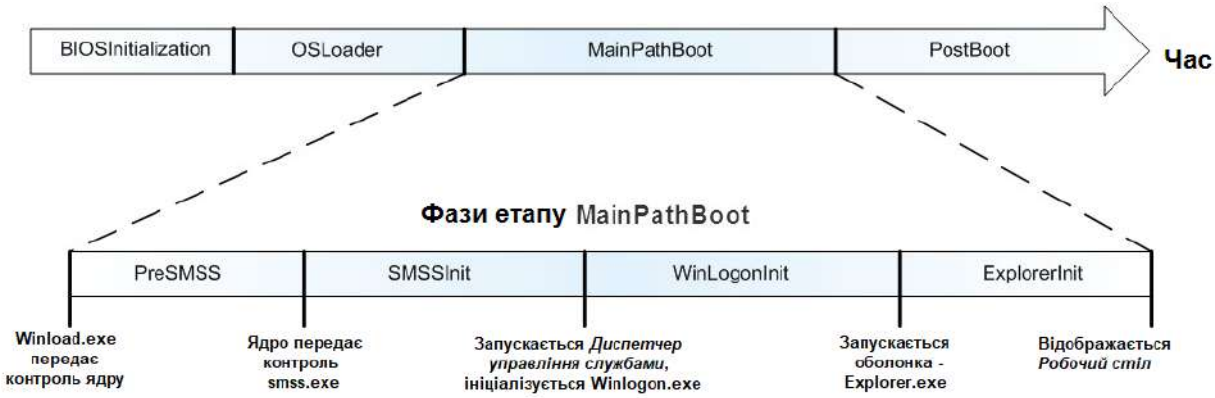


Рис. 12.14. **Фази етапів завантаження операційної системи**

На етапі **OSLoader**

завантажувач Windows (**winload.exe**) завантажує основні системні драйвери, необхідні для прочитування мінімально необхідного набору даних із диска;

потім завантажувач ініціалізував систему до моменту, із якого стає можливим завантаження ядра;

коли ядро починає завантажуватися, **winload.exe** поміщає в оперативну пам'ять системний розділ реєстру і додаткові драйвери, позначені як BOOT_START.

Візуально етап **MainPathBoot** починається з екрана "Завантаження Windows" і завершується під час появи Робочого столу. Якщо не налаштовано автоматичний вхід до системи, тривалість цього етапу збільшується за рахунок часу, який потрібно для введення пароля.

Під час етапу MainPathBoot відбувається основна робота із завантаження операційної системи: ініціалізувалося ядро; відбувається визначення пристроїв Plug and Play (PNP); запускаються служби; виконується вхід до системи; ініціалізувався Explorer, тобто система готується до завантаження Робочого столу.

Етап складається із чотирьох фаз, кожна з яких володіє власними характеристиками та може по-своєму впливати на тривалість завантаження системи.

Фаза **PRESMSS** (у графічному зображенні WPT її позначено як **Pre Session Init**) починається з ініціалізації ядра.

Під час неї

ядро ініціалізувало структури даних і компоненти, а потім запускає диспетчер PNP;

диспетчер PNP, своєю чергою, ініціалізував драйвери BOOT_ START, які було завантажено за допомогою **winload.exe** на етапі **OSLoader**;

коли диспетчер PNP виявляє пристрій, він завантажує необхідний драйвер і виконує його ініціалізацію.

Візуально початок фази **SMSSInit** неможливо визначити. Її частиною є порожній екран, який відображається між заставкою та екраном входу до системи, чия поява сигналізує про завершення фази.

Фаза **SMSSInit** (у графічному зображенні WPT її позначено як **Session Init**) починається з того, що ядро передає контроль диспетчерові сесій (**smss.exe**).

Під час цієї фази система

ініціалізувала реєстр;

завантажує і запускає пристрої та другу хвилю драйверів, які не позначено як BOOT_START;

запускає процеси підсистеми;

Фаза завершується з передаванням контролю над процесом **winlogon.exe**.

Майже кожен користувач Windows чув про так званий "**синій екран смерті" (**Blue Screen of Depth) або навіть бачив його. Цим зловісним терміном називають синій екран, що виникає під час припинення роботи Windows із-за катастрофічного перебою або внутрішньої помилки, що перешкоджає роботі системи.

Часто такою причиною стає посилання на адресу в пам'яті, що порушує права доступу, наприклад спроба запису на сторінку, яка має атрибут "тільки для читання", або спроба читання за ще не відображеною на пам'ять адресою. Іншою поширеною причиною є непередбачені вимкнення або переривання. Перебої також трапляються, коли одна з підсистем ядра (така як диспетчер пам'яті чи диспетчер електроживлення) або драйвер (наприклад, USB-драйвер чи драйвер дисплея) виявляють непогодженість виконуваних ним операцій.

Через яку б причину не відбувся перебій системи, у всіх випадках вона викликається функцією KeBugCheckEx. Ця функція приймає стоп-код (stop code), або контрольний **код помилки** (bugcheck code), і чотири параметри, які інтерпретуються, залежно від цього коду.

На основі даних, зібраних у Windows версій із 7 до 7 SP1, двадцять стоп-кодів, що найчастіше зустрічаються, які є причиною 91 % системних відмов, можна розподілити на такі категорії:

1. Звернення до відсутньої сторінки. Стоп-коди в цьому разі такі:

0xA – IRQL_NOT_LESS_OR_EQUAL;

0xD1 – DRIVER_IRQL_NOT_LESS_OR_EQUAL.

2. Управління електроживленням стоп-кодів:

0x9F – DRIVER_POWER_STATE_FAILURE.

3. Виключення і системні переривання. Стоп-коди такі:

0x1E – KMODE_EXCEPTION_NOT_HANDLED;

0x3B – SYSTEM_SERVICE_EXCEPTION;

0x7E – SYSTEM_THREAD_EXCEPTION_NOT_HANDLED;

0x7F – UNEXPECTED_KERNEL_MODE_TRAP;

0x8E – KERNEL_MODE_EXCEPTION_NOT_HANDLED із параметром P1, що недорівнює 0xC0000005 STATUS_ACCESS_VIOLATION.

4. Порушення прав доступу. Стоп-коди такі:

0x50 – PAGE_FAULT_IN_NONPAGED_AREA;

0x8E – KERNEL_MODE_EXCEPTION_NOT_HANDLED із параметром P1, що дорівнює 0xC0000005 STATUS_ACCESS_VIOLATION.

5. Дисплей. Стоп-код такий:

0x116 – VIDEO_TDR_FAILURE.

6. Пул. Стоп-коди такі:

0x19 – BAD_POOL_HEADER;

0xC2 – BAD_POOL_CALLER;

0xC5 – DRIVER_CORRUPTED_EXPOOL.

7. Управління пам'яттю. Стоп-коди такі:

0x1A – MEMORY_MANAGEMENT;

0x4E – PFN_LIST_CORRUPT.

8. Апаратне забезпечення таке:

0x7A – KERNEL_DATA_INPAGE_ERROR;

0x124 – WHEA_UNCORRECTABLE_ERROR.

9. USB. Стоп-код такий:

0xFE – BUGCODE_USB_DRIVER.

10. Важливий об'єкт. Стоп-код такий:

0xF4 – CRITICAL_OBJECT_ TERMINATION.

11. Файлова система NTFS. Стоп-код такий:

0x24 – NTFS_FILE_ SYSTEM.

За замовуванням усі системи сім'ї Windows налаштовано на запис інформації про стан на момент перебою.

Під час завантаження система дістає параметри аварійного дампа з розділу HKLM\SYSTEM\CurrentControlSet\Control\CrashControl реєстру. Якщо генерацію дампа задано, створюється копія драйвера міні-порту диска, через яку том записується в пам'ять. Цій копії надають ім'я міні-порту із префіксом dump.

Різні перебої можна викликати за допомогою програми Notmyfault, створеної у Windows Sysinternals. Вона складається з виконуваного файла Notmyfault.exe і драйвера Myfault.sys. Після запуску цей файл завантажує драйвер і виводить показане на рис. 12.15 діалогове вікно, що дозволяє різними способами припинити роботу систему або викликати її зависання, а також змусити драйвер ініціювати витік пам'яті з вивантажуваного або невивантажного пулу.

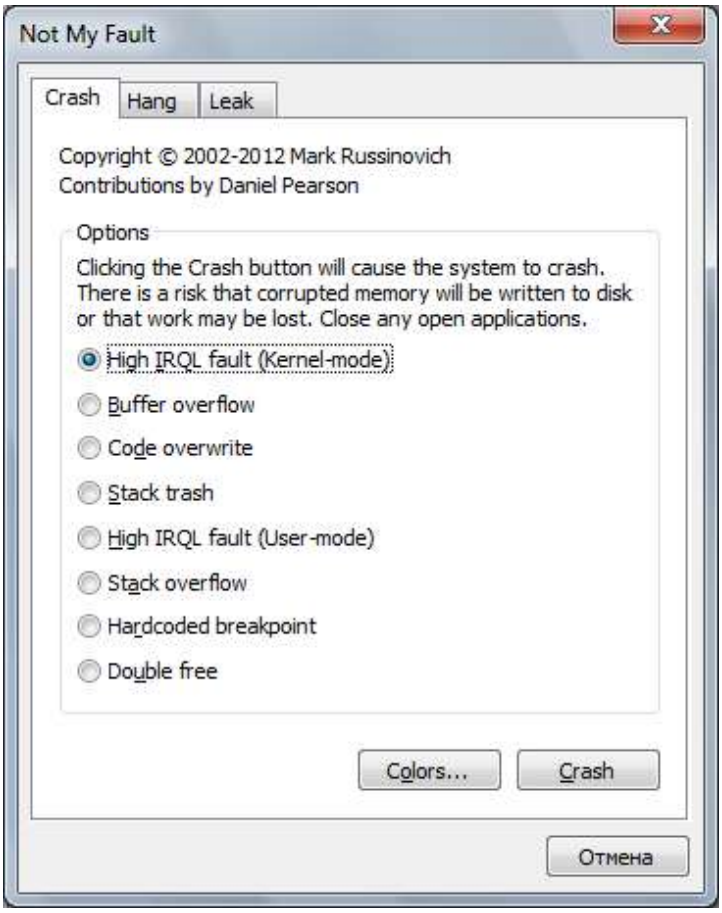


Рис. 12.15. **Діалогове вікно вибору причини перебою в роботі ОС**

Найпростіший перебій у додатку Notmyfault викликано встановленням перемикача High IRQL Fault (Kernel-Mode) і клацанням на кнопці Crash.

Після цього драйвер виділяє сторінку в купі підкачуваної пам'яті, збільшує IRQL-рівень до DPC/dispatch і звертається до звільненої сторінки. Якщо це не призводить до перебою, процес продовжує прочитувати пам'ять після кінця сторінки, поки не відбудеться перебій із-за звернення до недійсної сторінки. У результаті драйвер виконує кілька неприпустимих операцій:

1. Посилається на пам'ять, яка йому не належить.

2. Звертається до пулу підкачуваної пам'яті за IRQL-рівня DPC/dispatch і вищого, що неприпустимо, оскільки на цих рівнях помилки сторінок не вирішуються.

3. Виходить за межі виділеної області пам'яті й намагається звернутися до пам'яті, яка потенційно може бути недійсною.

Перше звернення до сторінки не завжди призводить до перебою, тому що звільнена драйвером сторінка може залишитися в системному робочому наборі (рис. 12.16).

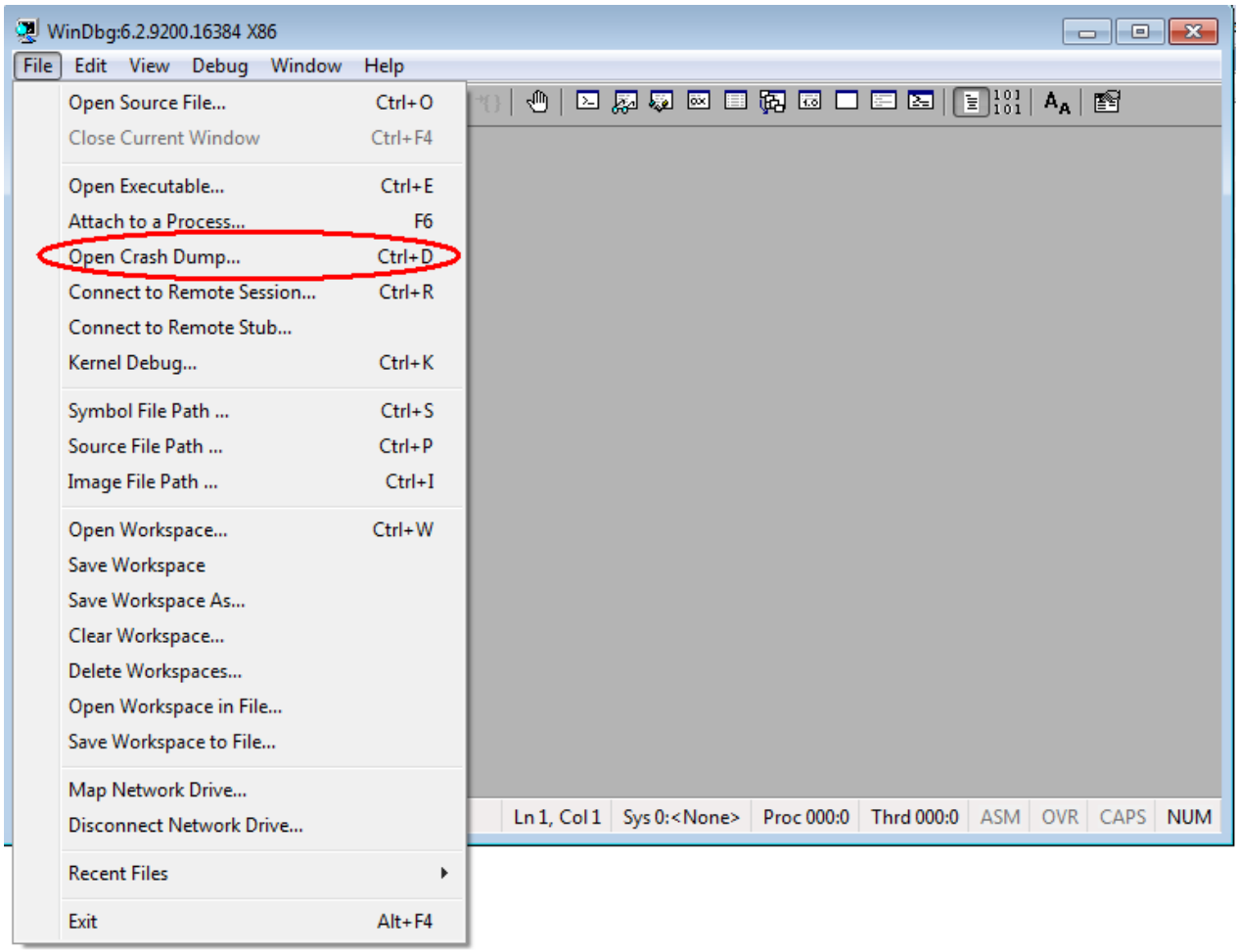


Рис. 12.16. **Аналіз дампу пам'яті у програмі WinDbg**

Після завантаження, що згенерованого цим перебоєм аварійного дампа в наладнику WinDbg, буде приблизно такий результат:

```
Microsoft (R) Windows Debugger Version 6.2.9200.16384 X86
Copyright (c) Microsoft Corporation. All rights reserved.
Loading Dump File [C:\Windows\Minidump\052614-18687-01.dmp]
Mini Kernel Dump File: Only registers and stack trace are available
Symbol search path is: *** Invalid ***
Executable search path is:
Unable to load image \SystemRoot\system32\ntkrnlpa.exe, Win32 error 0n2
*** WARNING: Unable to verify timestamp for ntkrnlpa.exe
*** ERROR: Module load completed but symbols could not be loaded for ntkrnlpa.exe
Windows 7 Kernel Version 7600 UP Free x86 compatible
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 7600.16385.x86fre.win7 rtm.090713-1255
Machine Name:
Kernel base = 0x82850000 PsLoadedModuleList = 0x82998810
Debug session time: Mon May 26 12:40:49.468 2014 (UTC + 4:00)
System Uptime: 0 days 0:00:42.484
Loading Kernel Symbols
.....................................................
.....................................................
Loading User Symbols
Loading unloaded module list
....
Unable to load image \??\C:\Windows\system32\drivers\myfault.sys, Win32 error 0n2
*** ERROR: Module load completed but symbols could not be loaded for myfault.sys
*******************************************************
*                        Bugcheck Analysis
*
```

```
*******************************************************************
Use !analyze -v to get detailed debugging information.
BugCheck D1 {93014008, 2, 0, 904dc5ab}
Probably caused by : myfault.sys ( myfault+5ab )
Followup: MachineOwner
---------
kd> !analyze -v
*******************************************************************
*                        Bugcheck Analysis
*******************************************************************
DRIVER_IRQL_NOT_LESS_OR_EQUAL (d1)
An attempt was made to access a pageable (or completely
invalid) address at an
interrupt request level (IRQL) that is too high.  This is
usually
caused by drivers using improper addresses.
If kernel debugger is available get stack backtrace.
Arguments:
Arg1: 93014008, memory referenced
Arg2: 00000002, IRQL
Arg3: 00000000, value 0 = read operation, 1 = write oper-
ation
Arg4: 904dc5ab, address which referenced memory
Debugging Details:
------------------
ADDITIONAL_DEBUG_TEXT:
You can run '.symfix; .reload' to try to fix the symbol
path and load symbols.
MODULE_NAME: myfault
FAULTING_MODULE: 82850000 nt
DEBUG_FLR_IMAGE_TIMESTAMP:  4f806ca0
READ_ADDRESS: unable to get nt!MmSpecialPoolStart
unable to get nt!MmSpecialPoolEnd
unable to get nt!MmPagedPoolEnd
unable to get nt!MmNonPagedPoolStart
unable to get nt!MmSizeOfNonPagedPoolInBytes
 93014008
```

```
CURRENT_IRQL:  0
FAULTING_IP:
myfault+5ab
904dc5ab 8b08            mov     ecx,dword ptr [eax]
CUSTOMER_CRASH_COUNT:  1
DEFAULT_BUCKET_ID:  WIN7_DRIVER_FAULT
BUGCHECK_STR:  0xD1
LAST_CONTROL_TRANSFER:  from 904dc5ab to 828967eb
STACK_TEXT:
WARNING: Stack unwind information not available. Follow-
ing frames may be wrong.
92926b3c 904dc5ab badb0d00 8412d240 93013008 nt+0x467eb
92926bb8   904dc9db   85d9a4f0   92926bfc   904dcb26   my-
fault+0x5ab
92926bc4   904dcb26   85d844a8   00000001   00000000   my-
fault+0x9db
92926bfc   8288c4bc   853289b0   85d9a4f0   85d9a4f0   my-
fault+0xb26
92926c14 82a8deee 85d844a8 85d9a4f0 85d9a560 nt+0x3c4bc
92926c34 82aaacd1 853289b0 85d844a8 00000000 nt+0x23deee
92926cd0 82aad4ac 853289b0 85d9a4f0 00000000 nt+0x25acd1
92926d04 8289342a 000000c4 00000000 00000000 nt+0x25d4ac
92926d34 777964f4 badb0d00 0020f270 00000000 nt+0x4342a
92926d38 badb0d00 0020f270 00000000 00000000 0x777964f4
92926d3c 0020f270 00000000 00000000 00000000 0xbadb0d00
92926d40 00000000 00000000 00000000 00000000 0x20f270
STACK_COMMAND:  kb
FOLLOWUP_IP:
myfault+5ab
904dc5ab 8b08            mov     ecx,dword ptr [eax]
SYMBOL_STACK_INDEX:  1
SYMBOL_NAME:  myfault+5ab
FOLLOWUP_NAME:  MachineOwner
IMAGE_NAME:  myfault.sys
BUCKET_ID:  WRONG_SYMBOLS
Followup: MachineOwner
---------
```

Windbg – це могутній наладник як для додатків, так і драйверів. Безліч плагінів, команд, скриптова мова. WinDbg можна використовувати як дебагер додатків призначеного для користувача режиму. Сам дебагер містить величезну кількість команд, але використовує дуже малу їхню частина. Докладніший опис можна знайти в рекомендованій літературі.

Загальне завдання для виконання

1. Аналітична частина

Етап 1. Зробіть аналіз головного завантажувального запису (MBR) жорсткого диска за допомогою програми Victoria. Результати, підтверджені скриншотами, занесіть до табл. 12.9.

Таблиця 12.9

Характеристики жорсткого диска за програмою Victoria

Standard					Advanced (Partition viewer)						
Model	Firmware	Serial	Disk size	CHS	N	Boot	System	Start LBA	End LBA	Size	Name
					1						
					...						

За допомогою програми BootIce, прочитавши MBR, подайте скриншот таблиці розділів жорсткого диска. Зробіть висновки про відмінності у значеннях однотипних параметрів різних програм.

Етап 2. Зробіть аналіз головного завантажувального запису (MBR) Flash-накопичувача за допомогою програми Victoria. Виконайте тестування свого Flash-накопичувача. Результати, підтверджені скриншотами, занесіть до табл. 12.10.

Таблиця 12.10

Характеристики Flash-накопичувача за програмою Victoria

Standard						
Model	Firmware	Serial	Disk size	CHS	Features	Sector

2. Дослідна частина

Етап 3. Проведіть дослідження елементів системного сховища конфігураційних даних (BCD store) для вказаного в п. 1 індивідуального варіанта задавання об'єкту BCD (табл. 12.11).

Таблиця 12.11

BCD-параметри для об'єкта {_____}

GUID об'єкта – { - - - - }			
Ім'я елемента	Константа	Значення	Призначення
device	11000001	partition = Z	пристрій завантаження
...	...	...	...

Етап 4. Проведіть дослідження фаз завантаження операційної системи. Результати, підтверджені скртншотами, занесіть до табл. 12.12.

Таблиця 12.12

Тривалість фаз завантаження Windows

№ фази	Ім'я фази	Початок, с	Закінчення, с	Тривалість, с
1	Pre session Init			
2	Session Init			
3	Winlogon Init			
4	Explorer Init			
5	Post Boot			

Етап 5. Виконайте визначення переліку завантажуваних драйверів, використовуючи меню Driver Delays, на етапі (фазі), заданій у п. 2 індивідуального завдання, час виконання яких не перевищує 400 ms. Результати, підтверджені скриншотами, занесіть до табл. 12.13, причому на скриншоті відобразіть **тільки** ті, що задовольняють цій умові драйвера.

Таблиця 12.13

Завантажувані драйвера під час фази ____

№ фази	Ім'я драйвера	Тривалість, с
1	Ntfs.sys	307
...	...	...

Етап 6. Зробіть базовий аналіз аварійного дампа, отриманого, унаслідок перебою, що штучно згенерував синій екран смерті. Використовуйте програму Notmyfault із драйвером myfault.sys, а причину відмов задайте, відповідно до п. 3 індивідуального варіанта завдання. Подайте скриншот BSOD, лістинг аварійного дампа (без автоматичних коментарів).

Індивідуальні завдання для виконання

Варіант № 1

1. {bootmgr}.
2. Ім'я фази: Pre session Init.
3. Причина BSOD: High IRQL fault (kernel-mode) – помилка високого IRQL (у режимі ядра).

Варіант № 2

1. {bootloadersettings}.
2. Ім'я фази: Session Init.
3. Причина BSOD: Buffer overflow (переповнювання буфера).

Варіант № 3

1. {globalsettings}.
2. Ім'я фази: Winlogon Init.
3. Причина BSOD: Code overwrite (помилка відкладеного запису).

Варіант № 4

1. {hypervisorsettings}.
2. Ім'я фази: Explorer Init.
3. Причина BSOD: Stack trash (помилка стека кошика).

Варіант № 5

1. {dbgsettings}.
2. Ім'я фази: Post Boot.
3. Причина BSOD: High IRQL fault (user-mode) – помилка високого IRQL (у режимі користувача).

Варіант № 6

1. {emssettings}.
2. Ім'я фази: Pre session Init.
3. Причина BSOD: Stack overflow (переповнювання стека).

Варіант № 7

1. {badmemory}.
2. Ім'я фази: Session Init.
3. Причина BSOD: hardcoded breakpoint (помилка захисту зупину).

Варіант № 8

1. {memdiag}.
2. Ім'я фази: Winlogon Init.
3. Причина BSOD: Double free.

Варіант № 9

1. {bootmgr}.
2. Ім'я фази: Explorer Init.
3. Причина BSOD: High IRQL fault (kernel-mode) – помилка високого IRQL (у режимі ядра).

Варіант № 10

1. {bootloadersettings}.
2. Ім'я фази: Post Boot.
3. Причина BSOD: Buffer overflow (переповнювання буфера).

Варіант № 11

1. {globalsettings}.
2. Ім'я фази: Pre session Init.
3. Причина BSOD: Code overwrite (помилка відкладеного запису).

Варіант № 12

1. {hypervisorsettings}.
2. Ім'я фази: Session Init.
3. Причина BSOD: Stack trash (помилка стека кошика).

Варіант № 13

1. {dbgsettings}.
2. Ім'я фази: Winlogon Init.
3. Причина BSOD: High IRQL fault (user-mode) – помилка високого IRQL (у режимі користувача).

Варіант № 14

1. {badmemory}.
2. Ім'я фази: Explorer Init.
3. Причина BSOD: Stack overflow (переповнювання стека).

Варіант № 15

1. {hypervisorsettings}.
2. Ім'я фази: Post Boot.
3. Причина BSOD: hardcoded breakpoint (помилка захисту зупину).

Варіант № 16

1. {globalsettings}.
2. Ім'я фази: Pre session Init.
3. Причина BSOD: Double free.

Варіант № 17

1. {dbgsettings}.
2. Ім'я фази: Session Init.
3. Причина BSOD: High IRQL fault (kernel-mode) – помилка високого IRQL (у режимі ядра).

Варіант № 18

1. {emssettings}.
2. Ім'я фази: Winlogon Init.
3. Причина BSOD: Buffer overflow (переповнювання буфера).

Варіант № 19

1. {bootmgr}.
2. Ім'я фази: Explorer Init.
3. Причина BSOD: Code overwrite (помилка відкладеного запису).

Варіант № 20

1. {bootloadersettings}.
2. Ім'я фази: Post Boot.
3. Причина BSOD: Stack trash (помилка стека кошика).

Варіант № 21

1. {hypervisorsettings}.
2. Ім'я фази: Pre session Init.
3. Причина BSOD: High IRQL fault (user-mode) – помилка високого IRQL (у режимі користувача).

Варіант № 22

1. {badmemory}.
2. Ім'я фази: Session Init.
3. Причина BSOD: Stack overflow (переповнювання стека).

Варіант № 23

1. {emssettings}.
2. Ім'я фази: Winlogon Init.
3. Причина BSOD: hardcoded breakpoint (помилка захисту зупину).

Варіант № 24

1. {memdiag}.
2. Ім'я фази: Explorer Init.
3. Причина BSOD: Double free.

Варіант № 25

1. {bootmgr}.
2. Ім'я фази: Post Boot.
3. Причина BSOD: High IRQL fault (kernel-mode) – помилка високого IRQL (у режимі ядра).

Варіант № 26

1. {bootloadersettings}.
2. Ім'я фази: Pre session Init.
3. Причина BSOD: Buffer overflow (переповнювання буфера).

Варіант № 27

1. {globalsettings}.
2. Ім'я фази: Session Init.
3. Причина BSOD: Code overwrite (помилка відкладеного запису).

Варіант № 28

1. {hypervisorsettings}.
2. Ім'я фази: Winlogon Init.
3. Причина BSOD: Stack trash (помилка стека кошика).

Варіант № 29

1. {dbgsettings}.
2. Ім'я фази: Explorer Init.
3. Причина BSOD: High IRQL fault (user-mode) – помилка високого IRQL (у режимі користувача).

Варіант № 30

1. {emssettings}.
2. Ім'я фази: Post Boot.
3. Причина BSOD: Stack overflow (переповнювання стека).

Контрольні запитання

1. Поясніть відмінність етапів завантаження систем на базі BIOS і UEFI?
2. Назвіть основні компоненти завантажувального процесу у Windows.

3. Яка структура прихованого (зарезервованого системою) диска? У яких випадках він може бути відсутнім?

4. Назвіть допустимі тимчасові інтервали етапів завантаження Windows.

5. Яку роль у завантаженні відіграють процеси smss.exe, csrss.exe, wininit.exe?

6. Яким чином у системному реєстрі задають автоматичний запуск програм за замовчуванням?

7. Як здійснюють протоколювання завантаження в безпечному режимі?

8. У якому файлі відображається і фіксується стан завантаження?

9. Чому у Windows трапляються перебої?

10. На яких розділах жорсткого диска розміщується BOOTMGR?

11. Що зберігається у сховищі конфігурації завантаження?

12. Назвіть імена BCDEDIT, які належать до другого рівня ієрархії сховища конфігурації завантаження.

13. За якою ознакою визначають належність типу об'єкта сховища?

14. Що зберігає в реєстрі кожен BCD-елемент?

15. Які розділи диску належать до системних?

16. Якою позначкою позначено в головному завантажувальному записі ознаку системного розділу?

17. Перелічіть обов'язкові умови для створення прихованого системного розділу Windows.

18. Яким чином реалізується CHS-адресація у файловій системі?

19. Назвіть можливі ознаки активності розділу жорсткого диску.

20. Назвіть обмеження для CHS-адресації. Чим їх обумовлено?

21. Які способи адресації використовують для дисків, ємність яких понад 7,8 ГБ?

22. Де зберігається набір утиліт, які входять до складу Windows Performance Toolkit?

23. Назвіть склад системних утиліт Windows Performance Toolkit та способи їхнього виклику.

24. Що таке "режим гебернації"?

25. Назвіть способи перезавантаження комп'ютера?

26. У чому полягає специфіка фази завантаження Pre Session Init?

27. Дайте характеристику способу виклику аварійного дампа в режимі Buffer overflow.

Використана і рекомендована література

Основна

1. Оліфер В. Г. Комп'ютерні мережі. Принципи, технології, протоколи / В. Г. Оліфер, Н. А. Оліфер. – Київ : Вища школа, 2006. – 544 с.

2. Побегайло А. П. Системное программирование в Windows / А. П. Побегайло. – Санкт-Петербург : БХВ-Петербург, 2006. – 1056 с.

3. Руссинович М. Внутреннее устройство Microsoft Windows. В 2-х ч. Ч.1 / М. Руссинович, Д. Соломон ; пер. с англ. – Санкт-Петербург : Питер, 2013. – 800 с.

4. Руссинович М. Внутреннее устройство Microsoft Windows. В 2-х ч. Ч.2. Основные подсистемы ОС / М. Руссинович, Д. Соломон, А. Ионеску ; пер. с англ. – Санкт-Петербург : Питер, 2014. – 672 с.

5. Таненбаум Е. Сучасні операційні системи / Е. Таненбаум. – Санкт-Петербург : Пітер, 2010. – 1120 с.

6. Шеховцов В. А. Операційні системи / В. А. Шеховцов. – Київ : Видавнича група BHV, 2005. – 576 с.

Додаткова

7. Бэкон Дж. Операционные системы / Дж. Бэкон, Т. Харрис. – Киев : Издат. группа BHV ; Санкт-Петербург : Питер, 2004. – 800 с.

8. Голубничий Д. Ю. Системне програмування і операційні системи : навч. посіб. У 2-х ч. / Д. Ю. Голубничий, В. Ф. Третяк. – Харків : Вид. ХДЕУ, 2004. – 192 с.

9. Голубничий Д. Ю. Системне програмування та операційні системи. навч. посіб. У 2-х ч. / Д. Ю. Голубничий, В. Ф. Третяк, С. В. Кавун. – Харків : Вид. ХНЕУ, 2005. – 264 с.

10. Джонсон М. Разработка приложений в среде Linux / М. Джонсон, Э. Троян : пер. с англ. – Москва : ООО "И. Д. "Вильямс", 2007. – 544 с.

11. Домашев А. В. Программирование алгоритмов защиты информации : учеб. пособ. / А. В. Домашев, В. О. Попов, Д. И. Правиков и др. – Москва : Нолидж, 2000. – С. 160–268.

12. Кокорева О. И. Реестр Windows XP / О. И. Кокорева. – Санкт-Петербург : БХВ-Петербург, 2004. – 560 с.

13. Маклин Й. Установка и настройка Windows 7. Учебный курс Microsoft / Й. Маклин. – Москва : Русская редакция, 2011. – 848 с.

14. Попов А. В. Введение в Windows PowerShell / А. В. Попов. – Санкт-Петербург : БХВ-Петербург, 2009. – 464 с.

15. Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows / Дж. Рихтер ; пер. с англ. – Санкт-Петербург : Питер, 2006. – 704 с.

16. Румянцев П. В. Азбука программирования в Win32 API / П. В. Румянцев. – Москва : Горячая линия – Телеком, 2001. – С. 258–269.

17. Румянцев П. В. Работа с файлами в Win32 / П. В. Румянцев. – Москва : Горячая линия – Телеком, 2002. – С. 92–157.

18. Руссинович М. Внутренне устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. Мастер-класс / М. Руссинович, Д. Соломон ; пер. с англ. – Москва : Издательско-торговый дом "Русская Редакция", 2005. – 992 с.

19. Саймон Р. Windows 2003 API. Энциклопедия программиста / Р. Саймон ; пер. с англ. – Киев : ООО "ДиасофтЮП", 2004. – 1088 с.

20. Секунов Н. Ю. Программирование на C++ в Linux / Н. Ю. Секунов. – Санкт-Петербург : БХВ-Петербург, 2004. – 368 с.

21. Солдатов В. П. Программирование драйверов Windows / В. П. Солдатов. – Москва : Бином, 2004. – 480 с.

22. Сорокина С. И. Программирование драйверов и систем безопасности : учеб. пособ. / С. И. Сорокина, А. Ю. Тихонов, А. Ю. Щербаков. – Санкт-Петербург : БХВ-Петербург, 2003. – 256 с.

23. Столингс В. Операционные системы / В. Столингс. – Москва : Вильямс, 2002. – 848 с.

Інформаційні ресурси

24. Енциклопедія сучасної культури, фольклору і субкультур. Операційна система ReactOS. – Режим доступу : http://lurkmore.to/ReactOS.

25. Объединенный Открытый Проект: сайт для настоящих компьютерщиков. – Режим доступа : http://www.openproj.ru.

26. Офіційний сайт розробників ОС KolibriOS. – Режим доступу : http://www.kolibrios.org.

27. Офіційний сайт розробників ОС QNX. – Режим доступу : http://www.qnx.com.

28. Офіційний сайт розробників ОС ReactOS. – Режим доступу : http://www.reactos.org.

29. Про QNX по-русски. – Режим доступа : http://qnx.org.ru.

30. Технологии будущего для реального времени. – Режим доступа : http://www.qnx-russia.ru.

Зміст

Вступ....3
Розділ 1. Архітектура операційних систем....4
Лабораторна робота 1. Дослідження операційної системи ReactOS....4
Лабораторна робота 2. Дослідження операційної системи KolibriOS....16
Лабораторна робота 3. Дослідження операційної системи QNX NEUTRINO....23
Розділ 2. Оперативна пам'ять, потоки та процеси....37
Лабораторна робота 4. Моделювання процесів в операційній системі....37
Лабораторна робота 5. Дослідження властивостей процесів і потоків....67
Лабораторна робота 6. Дослідження властивостей віртуальної пам'яті....96
Розділ 3. Файлова система....141
Лабораторна робота 7. Дослідження виконуваного файла Windows....141
Лабораторна робота 8. Дослідження бібліотек динамічного компонування....155
Лабораторна робота 9. Дослідження системного реєстру ОС Windows....177
Розділ 4. Мережеві, багатопроцесорні операційні системи та захист інформації....210
Лабораторна робота 10. Дослідження системних служб і драйверів....210
Лабораторна робота 11. Дослідження засобів захисту даних....227
Лабораторна робота 12. Дослідження та оптимізація завантаження ОС Windows....280
Використана і рекомендована література....313
Основна....313
Додаткова....313
Інформаційні ресурси....314

НАВЧАЛЬНЕ ВИДАННЯ

Голубничий Дмитро Юрійович
Холодкова Анна Валеріївна

ОПЕРАЦІЙНІ СИСТЕМИ

Навчальний посібник

Самостійне електронне текстове мережеве видання

Відповідальний за видання *О. Г. Руденко*

Відповідальний редактор *М. М. Оленич*

Редактор *О. Г. Доценко*

Коректор *О. Г. Доценко*

План 2018 р. Поз. № 22-ЕНП. Обсяг 317 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
ДК № 4853 від 20.02.2015 р.