МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Д. Ю. Голубничий, А. В. Холодкова, О. В. Шматко, М. М. Козуля

ОПЕРАЦІЙНІ СИСТЕМИ

Лабораторний практикум
для студентів спеціальності 122 «Комп'ютерні науки»
та 121 «Програмна інженерія»
усіх форм навчання

Рекомендовано вченою радою НТУ «ХПІ»

Харків
НТУ «ХПІ»
2019

УДК 004.45
О60

Рецензенти:
В.О. Алексієв, д-р техн. наук, проф.,
Харківський національний економічний університет імені Семена Кузнеця;
І.В. Рубан, д-р техн. наук, проф.,
Харківський Національний університет радіоелектроніки

Рекомендовано вченою радою НТУ «ХПІ»
як лабораторний практикум для студентів комп'ютерних наук,
протокол № 3 від 01.03.2019 р.

Авторський колектив: Д. Ю. Голубничий к.т.н., доц., А. В. Холодкова к.т.н., доц., О. В. Шматко к.т.н., доц., М. М. Козуля к.т.н., доц.

О60 Операційні системи : лабораторний практикум для студентів спеціальностей 122 «Комп'ютерні науки» та 121 «Програмна інженерія» / Д. Ю. Голубничий, А. В. Холодкова, О. В. Шматко, М. М. Козуля. – Харків : НТУ «ХПІ», 2019. – 336 с.

ISBN 978-617-7602-65-0

Наводяться 12 лабораторних робіт, де розглядаються методи використання теорії побудови операційних систем для набуття студентами професійних умінь (компетенцій) та практичних навичок при дослідженні властивостей компонентів операційних систем. Наведено багато програмних продуктів, які ілюструють реалізацію конкретних завдань.

Призначено для студентів, викладачів і користувачів, які вивчають основи побудови операційних систем та їх практичне застосування при створенні додатків для різних галузей.

Іл. 165. Табл. 80. Бібліогр. 20 назв.

УДК 004.45

ISBN 978-617-7602-65-0

ЗМІСТ

ВСТУП

Лабораторне заняття – форма навчальної роботи, де студенти під керівництвом викладача особисто проводять натурні або імітаційні експерименти чи досліди з 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 дисципліни «Операційні системи» проводяться в спеціально обладнаному навчальному класі з використанням комп'ютерного обладнання, пристосованого до начального процесу.

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

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

З метою підвищення якості навчального процесу під час проведення лабораторного заняття навчальна група поділяється на дві підгрупи. Кожне студент працює самостійно, виконуючи загальні та індивідуальні завдання для лабораторного дослідження.

Перед виконанням лабораторної роботи викладачем проводиться вхідний контроль за навчальними питаннями з теми лабораторної роботи, які були розглянуті теоретично на попередніх лекціях. Це поточне оцінювання здійснюється під

час проведення лабораторних занять і має на меті перевірку рівня підготовленості студента до виконання конкретної роботи. За результатами виконаної на занятті лабораторної роботи студенти оформлюють індивідуальні звіти з її виконання та захищають перед викладачем.

Практичний модульний контроль проводиться після виконаних лабораторних завдань в межах відповідного модулю з урахуванням захищених звітів з лабораторних робіт.

Необхідним елементом успішного засвоєння навчального матеріалу дисципліни є самостійна робота студентів з науково-технічною літературою та з сучасними технічними засобами комп'ютеризованої обробки інформації.

Лабораторний практикум містить 12 лабораторних робіт, які присвячені дослідженню: операційних систем ReactOS, KolibriOS, QNX Neutrino; властивостей процесів і потоків; властивостей віртуальної пам'яті; бібліотек динамічного компонування; ситемного реєстру ОС Windows; засобів захисту даних; оптимізації завантаження ОС Windows.

Лабораторна робота 1

ДОСЛІДЖЕННЯ ОПЕРАЦІЙНОЇ СИСТЕМИ REACTOS

Мета роботи: ознайомитись з основними принципами установки (інсталяції) операційної системи (ОС) ReactOS і її конфігурації на віртуальних машинних середовищах, а також набуття практичних навичок побудови і дослідження ReactOS у процесі інсталяції.

Указівки з підготовки до виконання лабораторної роботи

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

Додаткову інформацію можна отримати за посиланнями:

1) www.reactos.org
2) www.openproj.ru
3) http://lurkmore.to/ReactOS

Теоретичні відомості

ReactOS – це спроба розробити клон Windows з відкритим початковим кодом. Як зразок для копіювання була вибрана Microsoft Windows NT 4.0. Перед розробниками була поставлена мета не просто створити середовище, в якому запускались не тільки Windows-додатки, але і була можливість 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 орієнтуються 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 їх додатки. У розробку готуються підсистеми Java, OS/2, DOS, POSIX в

майбутньому, можливо, інші. ReactOS завжди планувалось розвивати за участі проекту WINE, щоб, по можливості, використовувати накопичений ними досвід та не переписувати одне й те саме. Здебільшого це стосується призначеної для користувача частини ОС, що приведе до повноти ядра ReactOS.

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

ReactOS підтримує файлові системи FAT12/16/32 та 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).

До проблем можна віднести 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 невисокі вимоги до апаратної конфігурації комп'ютера порівняно з ОС клона Windows. Її можна тільки встановити на апаратній платформі x86.

Мінімальні вимоги до x86 комп'ютера для інсталяції ReactOS, наступні:

1. Мінімальна частота процесора – 600 MHz.
2. Рекомендована частота процесора – 1.6 GHz.
3. Мінімальний об'єм ОЗУ – 128 MB.
4. Рекомендований об'єм ОЗУ – 256 MB.

5. Вільний дисковий простір – 2 Гб.
6. Вимоги до монітора:

– діагональ – не нижче 17’;

– мінімальна роздільна здатність монітора – 1024 x 768;

– глибина палітри – 65535 кольорів.

7. Наявність маніпулятора – «миша».
8. Клавіатура – стандартна.

Етап 2. Створення нової віртуальної машини в VMWare

Дистрибутив операційної системи ReactOS має розмір близько 50 Мбайт. Оскільки в дистрибутиві міститься тільки найнеобхідніші компоненти – ядро, бібліотеки і декілька стандартних додатків, під час створення нової віртуальної машини в розділі вказівки ***типу ОС***, яка буде інстальована, необхідно вказати Windows NT, тому що скільки 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. На рис. 1.1–1.7 наведені скріншоти інсталяції системи ReactOS на емуляторі VMware.

Етап 3. Первинне налаштування установки (інсталяції) ReactOS

Після підключення ReactOS.iso та запуску віртуальної машини з’являється початкова текстова сторінка установки, яка надана на рис. 1.1.

На даному етапі встановлюються налаштування монітора, клавіатури (українська розкладка відсутня), тип комп’ютера (рис. 1.2).

Етап 4. Вторинне налаштування установки (інсталяції) ReactOS

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

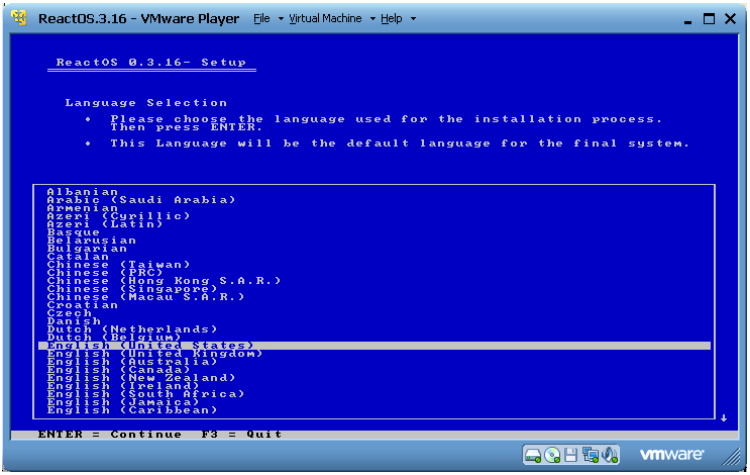


Рисунок 1.1 – Вибір мови системи за умовчанням

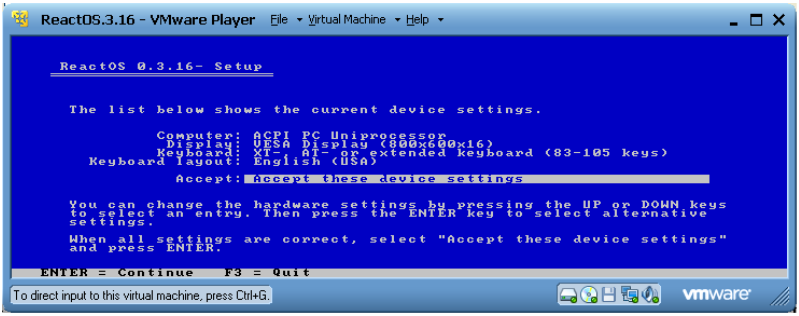


Рисунок 1.2 – Вікно задання налаштувань пристроїв

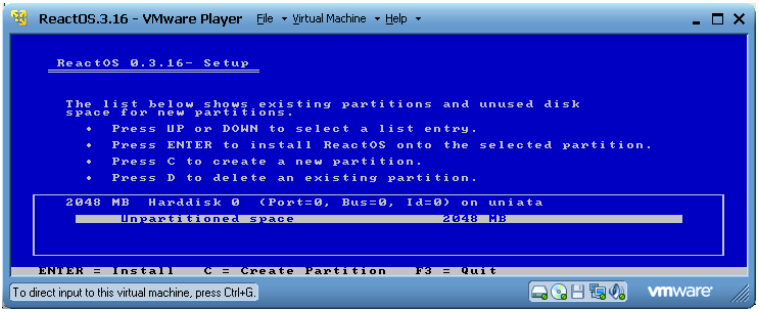


Рисунок 1.3 – Вікно вибору диска, на який буде проведена інсталяція

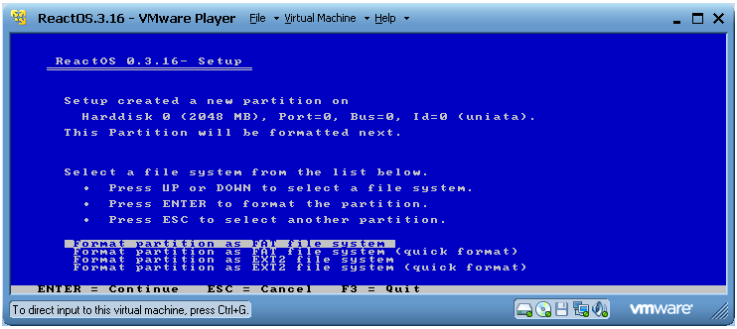


Рисунок 1.4 – Вікно вибору типу файлової системи

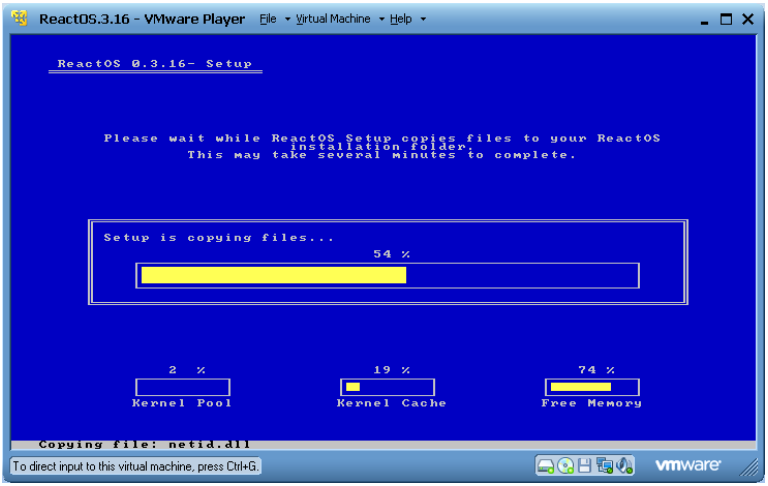


Рисунок 1.5 – Вікно індикації копіювання файлів інсталятора

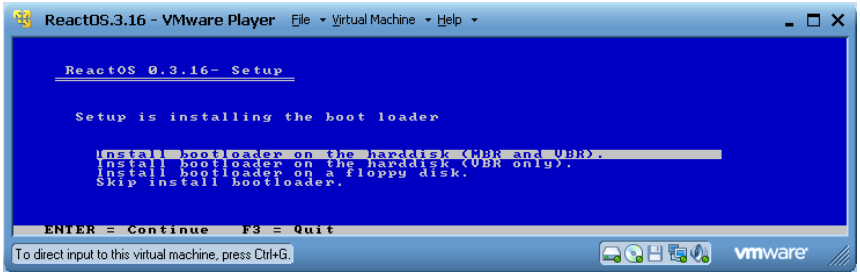


Рисунок 1.6 – Вікно вибору місця інсталяції завантажувального сектора

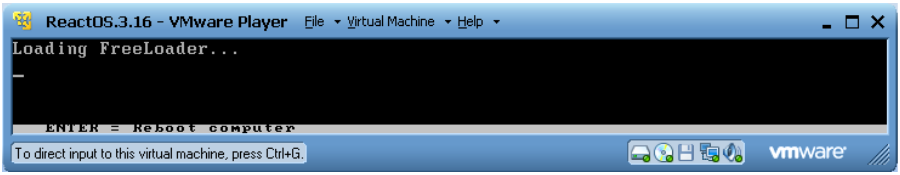


Рисунок 1.7 – Завершення первинної настройки інсталяції й автоматичне перезавантаження системи

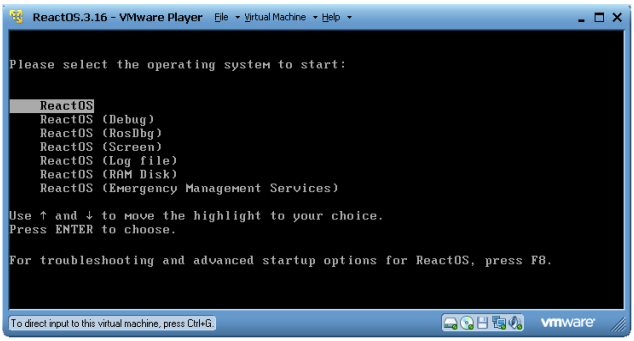


Рисунок 1.8 – Вибір варіанта завантаження системи

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

У цілях безпечного проведення установлення незнайомої і недобробленої системи запит був відхилений. І нарешті, після вказівки імені користувача – адміністратора, його пароля, імені комп'ютера і організації, а також вибору налаштування мов, дати, часу і часового поясу проводиться первинне завантаження Робочого столу операційної системи ReactOS (рис.1.10).

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

- 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).

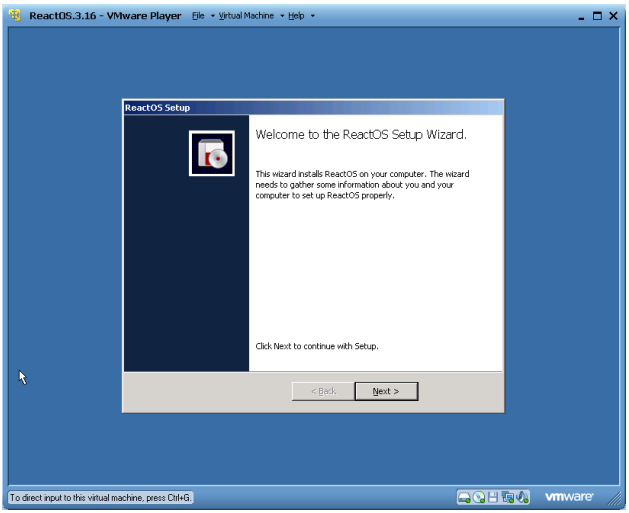


Рисунок 1.9 – Майстер установки ReactOS для персонального налаштування

Рисунок 1.10 – Робочий стіл системи ReactOS

У системі присутні ігри – аналоги Сапера, Павука і Пасьянсу у Windows. Багато елементів інтерфейсу не допрацьовано – є тільки форма, а функції, які повинні їм відповідати, ще не реалізовані.

На цьому установлення завершується.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Таблиця 1.1 – Індивідуальні завдання для виконання

Варіант	Команди інтерпретатора		Розділ	Обчислення
1.	CMD	POPD	Games & Fun	суми позитивних елементів масиву
2.	RMDIR	HISTORY	Finance	суми елементів масиву, розташованих між першим і останнім негативними елементами
3.	RENAME	CLS	Engineering	добуток елементів масиву, розташованих між максимальним і мінімальним елементами
4.	ECHO	MKLINK	Graphics	максимальний елемент масиву
5.	CHCP	START	Office	суми елементів масиву з непарними номерами
6.	GOTO	VERIFY	Internet & Network	суми елементів масиву, розташованих між першим і останнім позитивними елементами
7.	ALIAS	LABEL	Development	номер максимального елемента масиву
8.	CALL	IF	Office	суми модулів елементів масиву, розташованих після мінімального за модулем елемента
9.	SET	CHOICE	Engineering	суми елементів масиву, розташованих після першого позитивного елемента
10.	PUSHD	SCREEN	Graphics	суми негативних елементів масиву
11.	BEEP	FOR	Games & Fun	суми 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 за 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	суми 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

Закінчення табл. 1.1

Варіант	Команди інтерпретатора		Розділ	Обчислення
22.	PATH	DATE	Office	суми цілих частин елементів масиву, розташованих після останнього негативного елемента
23.	ATTRIB	CD	Games & Fun	суми позитивних елементів масиву, розташованих до максимального елемента
24.	DIR	TYPE	Engineering	кількість елементів масиву, менших за число K
25.	MKDIR	FOR	Graphics	суми елементів масиву, розташованих до мінімального елемента
26.	VOL	PROMPT	Games & Fun	суми елементів масиву, розташованих до останнього позитивного елемента
27.	REPLACE	TIME	Internet & Network	суми 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

Хід роботи

1. Розглянути 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. Дослідити різні варіанти розширеного завантаження ReactOS при натисненні клавіші **F8**:

- безпечний режим роботи ОС, що обмежує доступ користувача до певних розділів системи, для забезпечення стабільності роботи;
- Safe Mode with Networking – безпечний режим, що надає доступ до мережевих можливостей комп’ютера;

- Safe Mode with Command Prompt – безпечний режим, що надає доступ до командного рядка;
- Enable Boot Logging – режим, що дозволяє провести протоколювання завантаження операційної системи;
- Enable VGA Mode – режим, що дозволяє перемикатися на інший графічний режим роботи;
- Last Known Good Configuration – режим, що дозволяє відновити систему до стану при останньому працездатному завантаженні;
- Directory Services Restore Mode (DSRM) – режим, призначений для відновлення Active Directory. Наприклад, коли Active Directory пошкоджений і потребує виправлення. Часто DSRM використовується для скидання забутих паролів користувачів і адміністратора домену;
- Debugging Mode – режим, що дозволяє запустити режим настройки операційної системи;
- Custom Boot – режим, що дозволяє вибрати інші способи завантаження системи, наприклад, завантаження вибраного диска, розділу системи, з файлу завантажувального сектора, вибраних компонентів ReactOS, завантаження ОС Linux, якщо відомий шлях до неї.

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

4. Провести установлення одного додатку (на особистий вибір студента) в ReactOS Application Manager з вказаного в таблиці 1.1 розділу завдання.

Наприклад. На рис. 1.11 проводиться установка браузера Mozilla Firefox в розділ Internet & Network.

Результати інсталяції відобразити в звітах з вказівкою алгоритму установлення, його особливостей і описом послідовності дій при виконанні кожного кроку завдань.

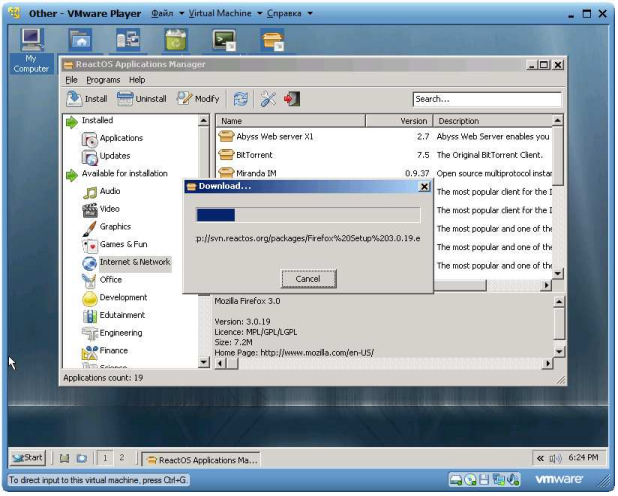


Рисунок 1.11 – Процес інсталяції браузера Mozilla Firefox

Додаткове завдання

5. Провести інсталяцію одного з вільно поширюваних компіляторів мови C++, наприклад, Dev-C++ (рис. 1.12).

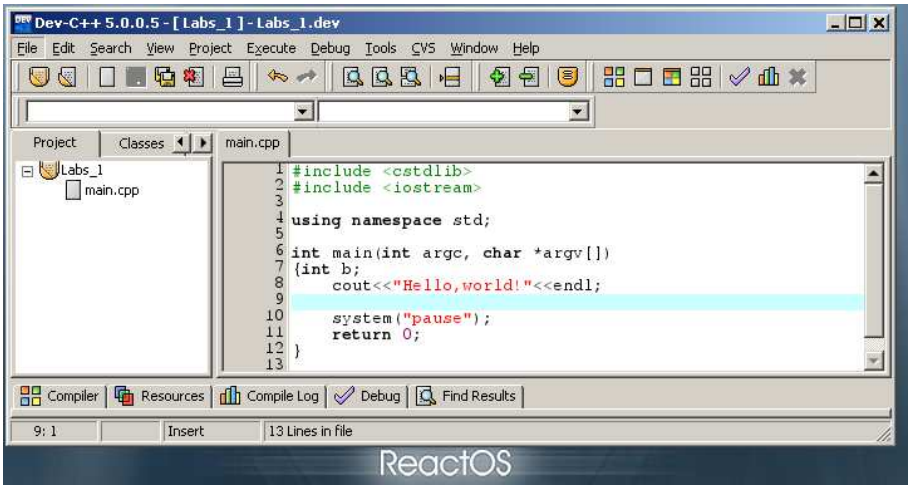


Рисунок 1.12 – Приклад створення програми в Dev-C++

6. Розробити програму виконання обчислення, яке задане в таблиці 1.1, з одновимірним масивом, що складається з *n* речових елементів.

Контрольні запитання

1. Визначите мету використання ОС.
2. У чому полягають принципові відмінності ОС ReactOS і версій 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.

Лабораторна робота 2

ДОСЛІДЖЕННЯ ОПЕРАЦІЙНОЇ СИСТЕМИ KOLIBRIOS

Мета роботи: ознайомитись з основними принципами установки (інсталяції) операційної системи KolibriOS і її конфігурації на віртуальних машинних середовищах та набуття практичних навичок побудови і дослідження KolibriOS у процесі інсталяції.

Вказівки по підготовці до виконання лабораторної роботи

Необхідно вивчити принципи і логіку роботи KolibriOS-інсталятора, його основні параметри при запуску в режимі командного рядка. Навчитися виконувати підготовчі операції на початковому етапі установки KolibriOS. Особливу увагу слід приділити процесу конфігурації ОС під час установки і після завершення кінцевого етапу. Вивчити вимоги до апаратного забезпечення ОС.

Додаткову інформацію для підготовки до роботи можна отримати в www.kolibrios.org.

Теоретичні відомості

KolibriOS (англ. KolibriOS) – операційна система для комп'ютера, яка повністю написана на асемблері fasm, поширювана на умовах GPL. Створена на основі MenuetOS. Є альтернативною операційною системою, оскільки вона використовує власні стандарти і не заснована на POSIX і UNIX. Система розрахована на використання асемблера для написання додатків, але є і програми, написані на мовах Ада, C, C++, Free Pascal, Forth. На даний момент переважна більшість розробників живуть на території країн колишнього СРСР. Це повноцінна операційна система з графічним інтерфейсом користувача з роздільною здатністю до 1280x1024 і 16 мільйонів кольорів, що включає більше 150 додатків (рис. 2.1).

KolibriOS може завантажуватись з дискети, CD диска і вінчестера. Для завантаження з вінчестера без використання дискети і зміни MBR (Master Boot Record – головного завантажувального запису) доводиться вдаватись до певних

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 з DOS, а в NT 5.x (і вище) потрібно використовувати файл boot.ini для вказівки завантажувального сектора. В цьому випадку в корені першого розділу повинен знаходитися файл образу ОС, і там же файл ядра kernel.mnt. Також існує завантажувач acroboot, який не набув великого поширення через інертність звичок користувачів.

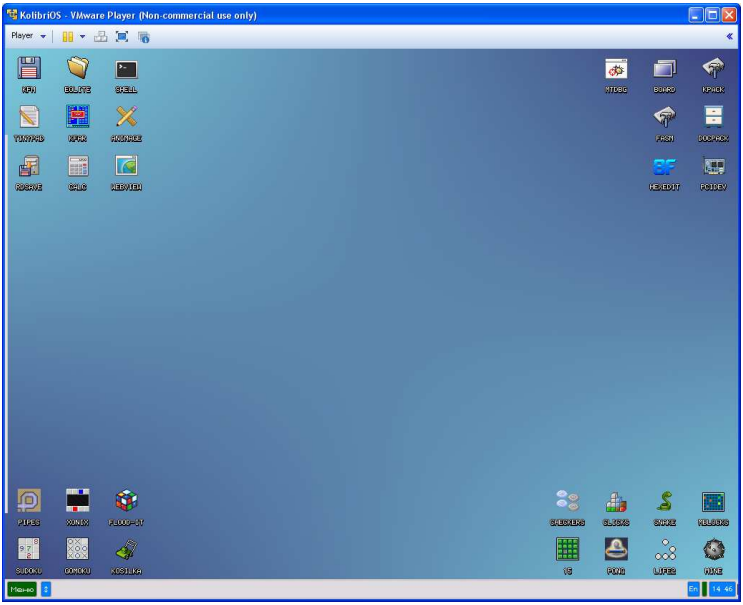


Рисунок 2.1 – Робочий стіл операційної системи KolibriOS

Ядро цілком написане на асемблері FASM. Завдяки чому досягається висока швидкодія і невеликий розмір системи.

Ядро монолітне, з можливістю завантаження зовнішніх драйверів (рис.2.2). На даний момент драйвери існують для підтримки звуку для кодеків AC97 (два варіанти: один драйвер для Intel і NVidia, інший – для SiS) і для підтримки

апаратного курсору миші у відеокарт ATI (для NVidia драйвер не існує).

Поняття процесу в KolibriOS на початковому рівні: процес – об'єднання потоків з одним і тим самим адресним простором. У всіх таких об'єднаних потоках одне й те саме ім'я і один й той самий розмір використовуваної пам'яті.

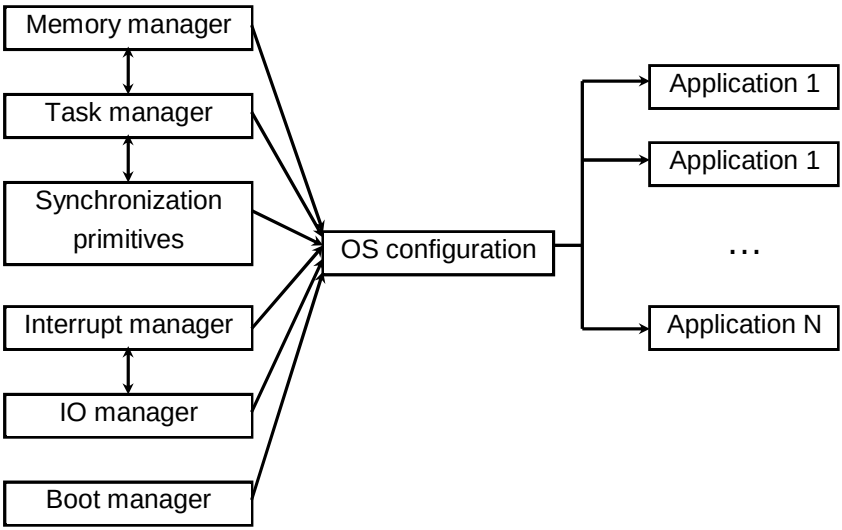


Рисунок 2.2 – Набір модулів ядра KolibriOS

Потоки існують і мають такі характеристики (memmap.inc з вихідних кодів ядра):

− ідентифікатор (TID), кожному створюваному потоку призначається унікальний ідентифікатор;

− стан потоку: активний (виконується прямо зараз або чекає перемикання завдань на нього), заморожений (завершується, той, що чекає);

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

− використання процесора: число тактів за останню секунду, яке процесор використав на виконання саме цього потоку;

− ім’я процесу (ім’я виконуваного файлу);

− маска подій, про які система сповіщає потік;

− системний стек;

− список об’єктів ядра, що асоціюються з цим потоком;

− карта дозволених портів введення/виведення;

− поточна папка для функцій файлової системи;

− буфер для повідомлень inter-process communication (IPC) (присутній, тільки якщо потік його явно визначив).

Процес не ідентифікується, інформація про потоки всередині ОС зібрана в статичний масив на 255 входів (нумерованих від 0 до 255, причому 0-й слот не може використовуватися, так що всього в системі може бути не більше 255 потоків). Деякі системні функції набувають номера слота, деякі – ідентифікатора.

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

Процесорний час ділиться між всіма потоками порівну, кожні 0,01 секунд 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 мишу (рисує курсор і відстежує події, що відбуваються з мишею 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 забиратись у ядро і/або писати драйвер. Для цього існує 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 і завантажитись 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; ввести kolibri.img, написати «a» і вставити чисту відформатовану дискету у віртуальний дисковод.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Таблиця 2.1 – Індивідуальні завдання для виконання

Вар.	**Назва гри**	**Demo**	**Обробка даних**	**Системний сенсор**	**Команди**		**Обчислення**
1	ArcanII	UnvWater	Hex-редактор	Диспетчер процесів	about	ls	$f(x) = x^2$
2	C4	Феєрверк	Калькулятор	PCI пристрій	alias	mkdir	$f(x) = ax$
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 disconnected	ls	cd	$f(x) = a/x$
12	SQ_Game	Кружок	Калькулятор	HDD інформатор	mkdir	alias	$f(x) = x^4$
13	Xonix	Шестерінки	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-лабиринт	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	$f(x) = x^2 + a$

Закінчення таблиці 2.1

Вар.	Назва гри	Demo	Обробка даних	Системний сенсор	Команди	Обчислення	Вар.
21	Новий Понг	Life	Hex-редактор	CPUID	alias	touch	$f(x) = a/x$
22	Падіння	Moveback	Калькулятор	GHOST монітор	about	rmdir	$f(x) = x^4$
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	Тетріс	Кружок	Tinypad	GHOST монітор	free	echo	$f(x) = x^3$
29	Трубопровід	Шестерінки	TextEdit	K.bus disconnected	history	cp	$f(x) = 1/x$
30	Шашки	Серце	Table	HDD інформатор	kill	ps	$f(x) = x^2 + x$

Хід роботи

1. Розглянути процес інсталяції KolibriOS на віртуальну машину.

2. Промоделювати різні варіанти завантаження системи. Продемонструвати скриншотами їх особливості:

- на підставі **iso-образу** з використанням CD/DVD-пристрою;
- на підставі **img-образу** з використанням віртуального floppy-пристрою.

3. Провести аналіз функціональних можливостей, умов запуску, порядок роботи (правила гри) з програмами KolibriOS з вказаного в таблиці 2.1 варіанта завдання.

4. Пояснити призначення, порядок запуску, розташування на RAM-носії і початковий асемблерний код демонстраційної програми, вказаної в таблиці 2.1.

5. Пояснити призначення, порядок використання програми обробки даних, що входять до складу KolibriOS, заданої в таблиці 2.1. Навести приклад обробки даних, підтвердивши відповідними скриншотами.

6. Провести аналіз системного сенсора відповідно до індивідуального варіанта завдання, вказаного в таблиці 2.1. Надати відповідні скріншоти з поясненнями вказаних у них параметрів.

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

Додаткове завдання

8. Провести інсталяцію одного з вільно-поширюваних компіляторів для KolibriOS.

9. Розробити програму виконання обчислення функції $f(x)$, яке задане в таблиці 2.1.

Контрольні запитання.

1. Визначите мету використання ОС KolibriOS.
2. У чому полягають принципові відмінності KolibriOS і клонів Windows?
3. Що дозволяє проявити використання служб ОС?
4. Назвіть основні принципи побудови ОС?
5. Укажіть призначення основних модулів ОС KolibriOS.
6. Назвіть основні версії KolibriOS.

Лабораторна робота 3

ДОСЛІДЖЕННЯ ОПЕРАЦІЙНОЇ СИСТЕМИ QNX NEUTRINO

Мета роботи: ознайомитись з основними принципами установки (інсталяції) операційної системи QNX і її конфігурації на віртуальних машинних середовищах, а також набуття практичних навичок побудови і дослідження QNX у процесі інсталяції.

Указівки з підготовки до виконання лабораторної роботи

Необхідно вивчити принципи і логіку роботи QNX-інсталятора, його основні параметри під час запуску в режимі командного рядка. Навчитися виконувати підготовчі операції на початковому етапі установки QNX. Особливу увагу слід приділити процесу конфігурації ОС під час інсталяції і після завершення кінцевого етапу. Вивчити вимоги до апаратного забезпечення ОС.

Додаткову інформацію можна отримати за посиланнями:

1) www.qnx.com.
2) www.qnx-russia.ru.
3) qnx.org.ru.

Теоретичні відомості

QNX (вимовляється як «кью-ен-ікс» або «кьюнікс») POSIX – сумісна операційна система реального часу, призначена переважно для вбудованих систем. Вважають за одну з кращих реалізацій концепції мікроядерних ОС.

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

Система невеликого обсягу, яка в мінімальній комплектації вміщується на одну дискету, разом з цим вона вважається швидкою і належним чином «закінченою» (що практично не містить помилок).

QNX Neutrino, випущена в 2001 році, перенесена на більшість платформ і зараз здатна працювати практично на будь-якому сучасному процесорі, використовуваному на ринку вбудованих систем. Серед цих платформ присутні сімейства x86, MIPS, PowerPC, а також спеціалізовані сімейства процесорів, такі, як SH-4, ARM, StrongARM і xScale.

Версія для некомерційного використання доступна для скачування на вебсайті 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: можливість стороннім розробникам вносити зміни до коду, безкоштовна для некомерційного використання і платна – для використання комерційного. Причому розробники не зобов'язані робити надбанням громадськості свої досягнення, а цілком можуть зберігати їх у складі закритих систем.

QNX Neutrino існує в двох варіантах – 16 розрядна версія і 32 розрядна. Остання здатна виконувати як 16 так і 32 розрядні застосування одночасно, для цього потрібно лише інсталювання двох відповідних системних бібліотек, що розділяються. Таким чином, фактично є 2 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 інсталяції QNX Neutrino – з інсталяційного CD, який у вигляді образу можна викачати з сайту виробника. Диск цей є завантажувальним, і тому ніяких додаткових маніпуляцій не вимагає.

Типова інсталяція установка в розділ QNX

Інсталяція QNX до розділу проводиться в текстовому консольному режимі.

Позитивна сторона установки – це правильна інсталяція, так, як і для будь-якої іншої OS. Але багато в чому переваги визначаються тим, для чого необхідно

використовувати систему. Інсталяція в розділ QNX працює з оригінальною файловою системою QNX, яка:

- стійка до відмов;
- істотно швидша.

Установлюємо в SETUP комп'ютера первинне завантаження з CD (рис.3.1).

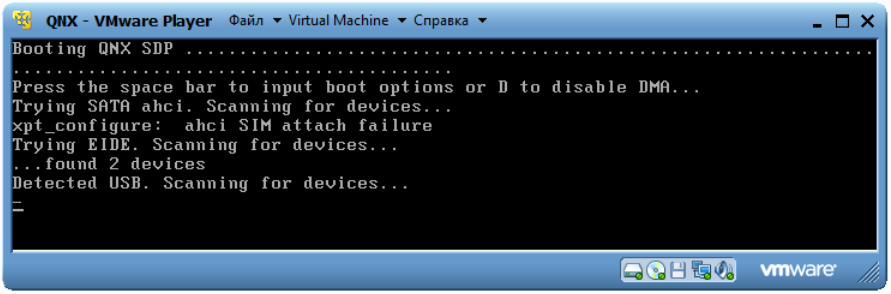


Рисунок 3.1 – Процес завантаження QNX, сканування пристроїв

Після завантаження користувачеві пропонується альтернативні можливості на вибір (названі тільки основні, реально їх більше: режим настройки, safe режим і т.д.) (рис.3.2). F2 – запуск з компакт-диска, F3 – інсталяція 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 ручну післяопераційну інсталяцію нового екземпляра на диск.

2. **F3 –** Завантаження на HDD QNX з CD, з підзавантаженням вже існуючої кореневої файлової системи QNX з диска. Дозволяє таким чином працювати з раніше створеним екземпляром системи (на диску може бути більше за один екземпляр кореневої файлової системи). Режим створення нового розділу QNX на диску з подальшою інсталяцією на нього системи QNX. Для того щоб скористатися цим (напівавтоматичним) режимом створення розділу і системи, потрібно перед-

бачити на диску нерозмічений простір необхідної величини, можливо, видаливши розділ, що раніше існував на ньому (наприклад, програмою fdisk Windows). Якщо буде вибраний режим створення розділу з інсталяцією, то система запропонує використовувати під створюваний розділ:

- весь вільний простір;
- половину вільного простору;
- чверть вільного простору і т.д.

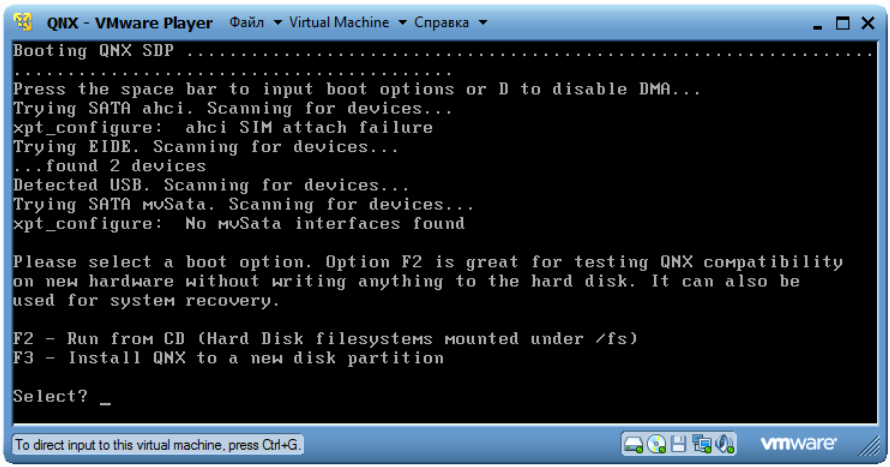


Рисунок 3.2 – Вибір варіанта завантаження

Після цього система:

- готує розділ для створення файлової системи (команда dinit);
- відзначає створений розділ як завантажувальний;
- записує початковий завантажувач QNX;
- копіює файл образу ядра .boot на розділ;
- копіює кореневу файлову систему QNX на розділ.

У процесі інсталяції системи користувача, так само, як при інсталяції в FAT32, запитує пароль користувача **root**, але можна залишити його порожнім

(рис.3.3). Додаткові реєстраційні записи користувача не створюються: це ще належить зробити пізніше вручну при адмініструванні створеної системи.

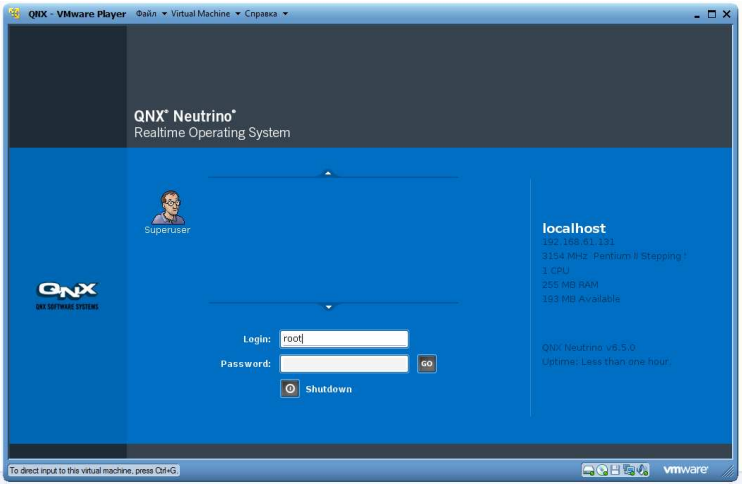


Рисунок 3.3 – Вікно введення пароля «root» адміністратора root

Наступне перезавантаження відбувається вже з розділу QNX під управлінням завантажувача QNX. При завантаженні він дозволяє вибрати розділ диска, з якого відбуватиметься завантаження, тобто вибрати одну зі встановлених на комп'ютері операційних систем (за наявності декількох дисків завантажувач QNX дозволяє вибрати і диск завантаження).

Інсталяція проходить у текстовому режимі. Починається воно з пропозиції створити дискові розділи, з чим можна погодитися, натиснувши Enter, або відмовити, натиснувши Esc, після чого відбувається перезавантаження. Можна також натиснути клавішу v (від англ. verbose), після чого всі подальші кроки супроводжуватимуться детальними коментарями (рис.3.4).

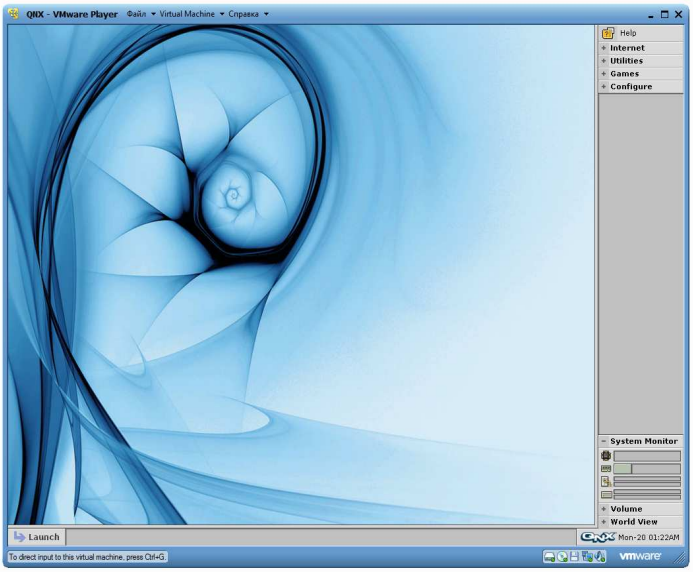


Рисунок 3.4 – Робочий стіл QNX Neutrino

Використання клавіатури в ОС QNX. При введенні 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>+<+>** – на наступну консоль;

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

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

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

Швидке перемикання шрифтів:

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

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

Кінець файлу – <**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 (програмне):

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

Основні команди в ОС QNX

При роботі користувача з ОС QNX використовується такий синтаксис команд (на прикладі команди **more**):

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

- a | -f – альтернатива опцій;

... – можливе повторення даного аргументу.

Перенаправлення введення-виведення

Більшість команд читає своє введення з файлу стандартного введення (**stdin**), яке зазвичай є клавіатурою і записує своє виведення до стандартного файлу виведення (**stdout**), який зазвичай є екраном дисплея. Проте, можливо:

- читати з файлу, відмінного від клавіатури;
- писати у файл, відмінний від екрана дисплея;
- читати з файлу або іншого пристрою < (введення перенаправлених символів);

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

– перенаправлення **stdout** у файл, додавання до файлу >> (додавання символів, що виводяться);

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

Наприклад, ps > /tmp/pr1

Отримання допомоги за конкретною командною проводиться 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 (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 {копіювати усі файли, окрім тих, які мають на кінці імені «о»}

Введення множини команд:

cp freg /tmp; ls /tmp

Робота у файловій системі ОС QNX

Ім’я файлу може бути до 48 символів у довжину.

Можливе застосування в іменах файлів символів з таким діапазонами:

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

Для переміщення програм бажано в іменах файлів використовувати символи, які визначені в **POSIX**: a...z; A...Z; 0 1 ... 9; _; .; -. Не можна використовувати символ «-» як перший символ імені файлу.

Суфікси в іменах файлів нічого не значать. У суфіксах є локальна угода (наприклад **.txt .c .h .o** і так далі).

Імена шляхів (маршрути)

/ – кореневий каталог (директорій);

//2/ – кореневий каталог на 2 вузлі;

.. – поточна директорія;

(/home/slava) – домашня директорія;

cd – повернення до домашнього каталогу.

Типова файлова система (ФС) QNX

У файловій системі ОС QNX зазвичай є такі директорії (каталоги), що зазначені у таблиці 3.1.

Таблиця 3.1 – Директорії (каталоги) файлової системі ОС QNX

Каталог	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 й 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 – друк імені робочої директорії;

ls [-l] – список змісту директорії.

Робота з файлами

diff – звіт про відмінність змісту двох файлів;

cat – конкатенація файлів, виведення вмісту файлу на екран;

cp – копіювання файлів;

wc – обхід структури директорії й виконання команд;

more – показ вмісту файлу в буферизованому вигляді;

less – інтерактивний (керований) перегляд виведення на екран;

lp – виведення змісту файлу на принтер;

mv – перенесення файлу;

rm – видалення файлу;

grep – пошук за рядком, заданим у вигляді шаблону;

sort – сортування, злиття або перевірка послідовності текстового файлу.

Інші команди

who – виведення на екран користувачів у системі;

ps – вивід на екран звіту про статус процесів;

sin – виведення системної інформації на екран.

find – пошук файлів.

write – відправлення повідомлення на термінал користувача.

wall – відправлення широкомовного повідомлення.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Таблиця 3.2 – Індивідуальні завдання для виконання

Вар.	Software	Команда		Обчислити
1	Web Browser	cat	uncompress	суму позитивних елементів масиву
2	Video Pocker	chgrp	sloginfo	суму 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	суму елементів масиву з непарними номерами
6	Solitaire	pwd	uesh	суму елементів масиву, розташованих між першим і останнім позитивними елементами
7	Snapshot	sh	su	номер максимального елемента масиву
8	Regiion View	csplit	stty	суму модулів елементів масиву, розташованих після мінімального за модулем елемента
9	Print Manager	dd	split	суму елементів масиву, розташованих після першого позитивного елемента
10	Peg	df	sloginfo	суму негативних елементів масиву
11	Othello	dispconf	netmanager	суму елементів масиву, розташованих між першим і другим негативними елементами
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 за 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	суму модулів елементів масиву, розташованих після першого негативного елемента
19	Dialer	shutdown	gzip	номер мінімального елемента масиву
20	Columns	hostname	who	кількість негативних елементів масиву

Закінчення табл. 3.2

Вар.	Software	Команда		Обчислити
21	Capture Screen	su	chmod	добуток елементів масиву, розташованих між першим і другим нульовими елементами
22	Calculator	ksh	df	суму цілих частин елементів масиву, розташованих після останнього негативного елемента
23	Terminal	cpio	slay	суму позитивних елементів масиву, розташованих до максимального елемента
24	Solitaire	ls	pax	кількість елементів масиву, менших за число K
25	Snapshot	mkdir	pidin	суму елементів масиву, розташованих до мінімального елементу
26	Regiion View	cp	ps	суму елементів масиву, розташованих до останнього позитивного елемента
27	Print Manager	more	pwd	суму елементів масиву, розташованих між першим і другим позитивними елементами
28	Peg	mount	rm	добуток позитивних елементів масиву
29	Othello	mv	script	номер максимального за модулем елемента масиву
30	Network	on	sendnto	добуток негативних елементів масиву

Хід роботи

1. Промоделювати різні варіанти завантаження системи:

− F2 – завантаження передвстановленої системи з використанням iso-образу на CD/DVD-пристрою;

− F3 – інсталяція системи на віртуальний жорсткий диск (необхідно введення ключа продукту).

2. Провести аналіз установленого програмного забезпечення, вказаного в таблиці 3.2 індивідуального завдання.

3. Визначити призначення, параметри виклику та приклад використання вбудованих типових команд інтерпретатора термінала відповідно до номера індивідуального варіанта (табл. 3.2).

4. Провести дослідження зареєстрованих у системі пристроїв. Результати занести до таблиці 3.3.

Таблиця 3.3 – Зареєстровані в системі пристрої

№ п/п	Назва пристрою	Значення інсталяції	Примітка
1	CD-ROM		
2	Клавіатура		
3	Миша		
4	Мережева плата		
5	Звуковий драйвер		
6	Дисплей		
7	Принтер		

Результати відобразити в звітах з ілюстрацією алгоритму інсталяції, її особливостей і описом послідовності дій при виконанні кожного кроку завдань.

Додаткове завдання

5. Розробити в QNX Photon Application Builder програму виконання в одновимірному масиві, що складається з *n* дійсних елементів, обчислення, яке задане у таблиці 3.2.

Контрольні запитання

1. Визначите мету використання ОС.
2. У чому полягають принципові відмінності QNX?
3. Що дозволяє проявити використання служб ОС?
4. Назвіть основні принципи побудови ОС.
5. Укажіть призначення основних модулів QNX Neutrino.
6. Назвіть основні версії QNX і місця їх використання.

Лабораторна робота 4

МОДЕЛЮВАННЯ ПРОЦЕСІВ В ОПЕРАЦІЙНІЙ СИСТЕМІ

Мета роботи: ознайомлення з основними принципами моделювання і дослідження функціонування ОС. Набуття практичних навичок моделювання процесів в ОС з досягненням максимальної продуктивності.

Вказівки з підготовки до виконання лабораторної роботи

Необхідно вивчити алгоритми планування, виконання процесів на процесорі, стратегії заміни сторінок (FIFO, LFU, NUR). Навчитись виконувати обчислення в табличному процесорі Excel. Особливу увагу приділити питанням синхронізації процесів з використанням механізму семафорів.

Додаткова інформація для підготовки до роботи:

1. Таненбаум Э. Современные операционные системы. – СПб.: Питер, 2010.
2. Шеховцов В.А. Операційні системи. – Київ: Видавнича група BHV, 2005.

Теоретичні відомості

У лабораторній роботі моделюються деякі аспекти функціонування ОС при обробці призначених для користувача і системних процесів. У роботі використовується програмна модель, що відображає такі аспекти функціонування ОС:

- поданняпроцесу в системі;
- взаємодія процесів у системі;
- виділення сторінок пам’яті під час запиту за допомогою використання алгоритмів заміщення сторінок;
- розподіл ресурсів обчислювальної системи між кількома процесами за допомогою механізму семафорів.

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

1. Три фіксовані пріоритети процесів – завдання користувача, серверні і системні завдання (завдання ядра).

2. Максимальна кількість призначених для користувача процесів – 10.

3. Кількість серверних процесів операційної системи – 9. В їх числі:

– менеджер пам’яті, що здійснює динамічне перетворення адресу;

– менеджер файлової системи (ФС), що обслуговує відповідні системні виклики.

4. Кількість системних завдань – 7. В їх числі:

– завдання, що обслуговує принтер;

– завдання, що обслуговує термінал;

– завдання, що обслуговує жорсткий диск (вінчестер);

– завдання, що обслуговує Flash-диск (дисковод);

– системний годинник;

– завдання, що обслуговує системні виклики (System Task) (взаємодія процесів користувача з ядром здійснюється без участі користувача);

– процес «апаратури», «порожній процес» – відбувається під час простою процесора.

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

Планувальник за умовчанням використовує дисципліну планування rabbit ring (RR), при цьому 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), довільно розбитий на сегменти коди, дані і стеки.

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

1. Керування процесами

Поточний стан віртуального комп’ютера, що надається користувачеві, нази-

вається **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 розміщується в основній пам'яті. Образ можна перенести на диск, якщо будь-якому пріоритетнішому процесу буде потрібне місце в основній пам'яті. Образ пам'яті складається з трьох сегментів (рис.4.1).

Сегмент повторновхідних процедур
Сегмент даних
Сегмент стека

Рисунок 4.1 – Образ пам'яті

Сегмент повторновхідних процедур може спільно використовуватися декількома процедурами – в первинній пам'яті такі сегменти, які розділяються, подаються єдиною копією. Сегментами процедур система керує централізовано за допомогою таблиці процедур (рис. 4.2).

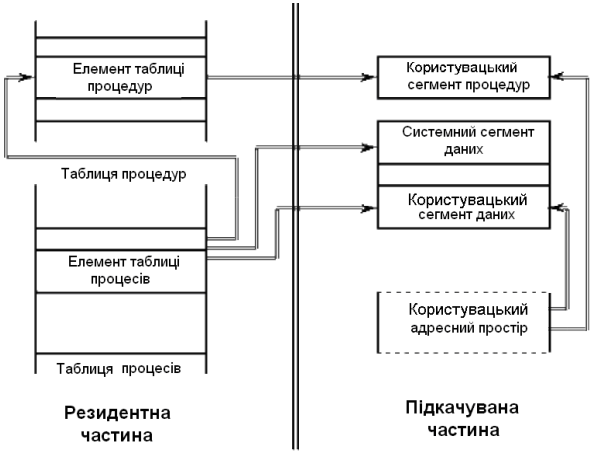


Рисунок 4.2 – Таблиці керування процесом

Кожен елемент таблиці містить адреси сегмента як у первинній, так і у вторинній пам'яті. Коли процес перший раз звертається до розподіленого сегмента, цей сегмент розміщується у вторинну пам'ять і заводиться елемент у таблиці процедур з відповідними адресами і лічильником, який вказує на число процесів, що спільно використовують цей сегмент. Коли в лічильнику опиниться нуль, звільняється елемент таблиці й ділянки первинної і вторинної пам'яті.

Сегмент даних розташовується одразу після сегмента процедур й може рости вниз. Він містить дані, записувані і зчитувані тільки одним процесом. Системні дані, що належать до процесу, зберігаються в окремому сегменті фіксованого розміру. Цей системний сегмент закінчується разом з процесом. Він містить:

- стан регістрів;
- дескриптори (описувачі) відкритих файлів;
- дані для розрахунків з використанням машини;
- область робочих даних;
- стек для системної фази виконання процесу.

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

Сегмент стека (див. рис. 4.1) починається зі старшої адреси у віртуальному адресному просторі і росте вгору в міру занесення до нього інформації при викликах підпрограм і перериваннях.

2. Синхронізація процесів

Синхронізація процесів здійснюється механізмом подій. Процеси чекають на події. Події подаються адресами відповідних таблиць. Батьківський процес, який чекає завершення одного з дочірніх процесів, чекає на події, подані адресою його власного елемента таблиці свого батьківського процесу.

3. Планування процесів

Процеси виконуються в двох станах: або у системному, або в 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 їх пріоритетів. Вищий пріоритет відданий процесу, який обробляє події і очікування. Події, пов'язані з роботою дисків, отримують наступний за старшинством пріоритет. Події, пов'язані з терміналами, часом доби і призначеними для користувача процесами, отримують відповідно нижчі пріоритети. Всі системні процеси мають вищий пріоритет. Призначені для користувача пріоритети залежать від часу завантаження процесора. Чим більше процес отримує призначеного для користувача часу, тим нижче його пріоритет. Такий метод планування забезпечує хороший час реакції під час діалогу.

4. Вштовхування-виштовхування

Процес може виштовхуватися у вторинну пам'ять і вштовхуватися зворотно, при цьому застосовується стратегія "перший відповідний". Якщо процес запрошує додаткову пам'ять, то йому буде виділено нову секцію пам'яті достатнього розміру і туди скопіюється вміст старої пам'яті. Стара пам'ять звільняється. Якщо у момент розширення процесу в первинній пам'яті не виявиться в наявності вільного блоку, процесу відводиться потрібне місце у вторинній пам'яті, він виштовхується і вштовхнеться знов у первинну пам'ять (вже враховуючи його новий розмір), коли в ній з'явиться достатній за розмірами вільний блок. Вштовхуванням і виштовхуванням керує спеціальний процес підкачок.

5. Конфігурація обчислювальної системи

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

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

мальний набір:

– термінал (TTY);
– жорсткий диск (Winchester);
– дисковод (Floppy);
– принтер (Printer).

Модель у своєму функціонуванні має саме таку конфігурацію.

6. Стан процесу

Призначений для користувача процес у даній моделі може знаходитися в одному з трьох станів:

– активний – у даний момент наданий ЦП;
– готовий (ready) – не має ЦП, але у будь-який момент часу може стати активним;
– блокований (unready) – не має ЦП і не може стати активним, оскільки чекає будь-якої події (наприклад, завершення операції введення/виведення). При настанні цієї події процес переходить в один з перших двох станів. На рис. 4.3 поданий перехід з одного стану в інший.

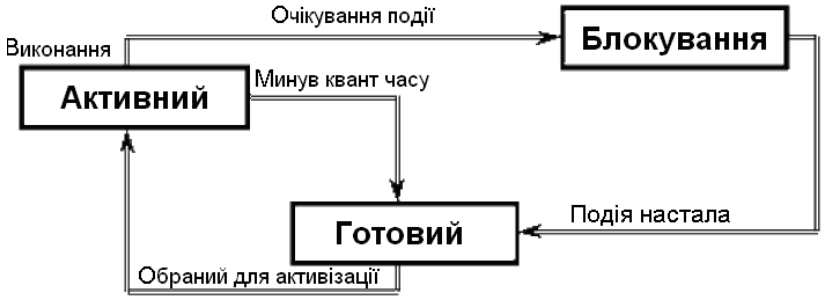


Рисунок 4.3 – Стани процесу

7. Взаємодія процесів у системі

У даній моделі призначені для користувача процеси взаємодіють з 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). Їх відмінності від призначених для користувача процесів у такому:

– серверні процеси мають вищий пріоритет;
– серверні процеси можуть безпосередньо взаємодіяти з системами;
– працюють з 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 даній моделі виключно важливу і складну роль, доцільно розглянути їх докладніше.

8. Керування пам'яттю (Memory Manager)

У моделі Memory Manager виконується функція динамічного перетворення адреси під час адресації віртуальної пам'яті (у UNIX трансляція адрес здійснюється безпосередньо ядром, а Memory Manager контролює динамічне використання пам'яті). Під час виконання Memory Manager на екрані з'являється вікно з картою оперативної пам'яті. При виникненні сторінкової відмови Memory Manager проводить завантаження необхідної сторінки, використовуючи поточну стратегію заміщення (вибирається користувачем). Можлива динамічна зміна розміру оперативної пам'яті в процесі моделювання.

9. Керування файлами (File System). Семафорні операції

File System містить двійкові семафори (*S*) для забезпечення взаємовиключення процесів при використанні пристроїв введення/виведення.

Коли процес звертається до якого-небудь пристрою введення/виводу, він виконує операцію *P* над відповідним семафором *P*(*S*) (операція *P* зменшує *S* на 1). Якщо при цьому стан семафора дорівнює 1, то він скидається, а процес отримує доступ до ПВВ, інакше процес переводиться в стан очікування (блокується).

Коли процес звільняє ресурс (наприклад, при завершенні операції введення/виведення), то, якщо є процеси що очікують ресурс, один з 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*). *CS* – частина процесу, під час якого деякий загальний ресурс повинен монополізуватися.

Виконання операцій *P* і *V* у даній моделі супроводжується появою на екрані відповідних вікон. Крім того, коли виконується процес файлової системи, на екрані з'являється вікно, що зображає список ПВВ обчислювальної системи і список процесів, що очікують до них доступ.

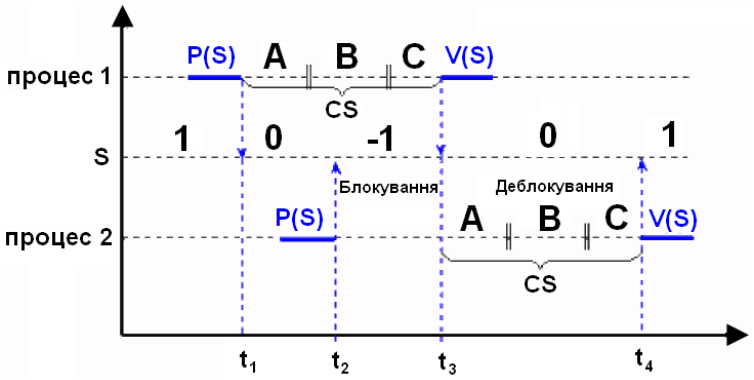


Рисунок 4.4 – Тимчасова діаграма двійкового семафора

Оскільки деякі ресурси використовуються процесами неподільно, то в моделі можливе виникнення тупиків (дедлоків), що відображає реальну проблему, яку необхідно розв'язувати при проектуванні ОС.

Поява вікна, що описує стан семафора, зупиняє роботу системи. Для продовження роботи потрібно натиснути будь-яку клавішу.

10. Призначені для користувача процеси

Типи системних викликів, які генеруються призначеними для користувача процесами, вибираються випадковим чином генератором випадкових чисел.

ПОРЯДОК ВИКОРИСТАННЯ МОДЕЛІ

1. Мета

Дана модель дозволяє вивчити деякі реальні механізми, які використовуються в багатьох операційних системах, а саме:

– розподіл ресурсів обчислювальної системи між декількома процесами за допомогою семафорів;

– взаємодія системних і прикладних завдань;

– подання процесу в системі (з точки зору ОС і самого процесу);

– різні рівні планування.

Модель дозволяє вивчити 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 отримати максимальну продуктивність системи.

2. Порядок використання моделі

Для функціонування моделі необхідні:

– персональний комп'ютер, сумісний з IBM PC;

– наявність кольорового монітора;

– операційна система Windows;

– достатнє для створення вихідного файлу 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 виконуються процеси трьох класів, що мають фіксований пріоритет: системні завдання (завдання ядра), серверні процеси і призначені для користувача процеси. Виділення процесам ресурсу ЦП здій-

снюється шляхом формування циклічних черг (*дисципліна RR*) запитів. При цьому процеси різних класів утворюють різні черги і планувальник кожного разу вибирає найбільш пріоритетний процес. Процеси в ході роботи формують запити до ресурсів. Взаємовиключення процесів при зверненні до ресурсів здійснюється за допомогою механізму семафорів. Для роботи процесам також необхідний ресурс оперативної пам'яті. ОП в моделі має сторінкову організацію. Якщо первинна пам'ять має брак, то для розміщення всіх процесів розділення відбувається шляхом виштовхування невикористовуваних сторінок у вторинну пам'ять.

Завдання ядра

Процеси цього класу є складовою частиною ядра системи і мають найвищий пріоритет. Дана модель включає такі задачі ядра:

– Завдання, що обслуговують ресурси (*Принтер, Термінал, Вінчестер, Дисковод*). Коли відбувається заняття або звільнення якого-небудь ресурсу система викликає відповідне обслуговуюче завдання. Тривалість роботи обслуговуючих завдань становить один квант модельного часу.

– Системний годинник. Це завдання, що підтримує системний час. Система викликає це завдання кожні 10 квант модельного часу. Тривалість роботи системного годинника складає один квант.

– System Task. Це завдання в реальній ОС обслуговує системні виклики, тобто здійснює взаємодію між призначеними для користувача процесами і ядром. Проте в даній моделі для спрощення взаємодія здійснюється 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 і завданнями ядра. Ці процеси такі, як і призначені для користувача, незалежні від ядра системи, проте мають переваги перед призначеними для користувача процесами. Серверні процеси можуть безпосередньо взаємодіяти з системою і працюють з реальним адресним простором. Серверні процеси мають більший пріоритет, чим призначені для користувача. У даній мо-

делі до серверних процесів належать ***Менеджер пам'яті***, що здійснює вштовхування/виштовхування сторінок пам'яті, і ***Файлова система***, що обслуговує доступ до ресурсів. Розглянемо їх докладніше.

Менеджер пам'яті

Якщо обсяг оперативної пам'яті, необхідної процесам, більший за наявну в обчислювальній системі, потрібно мати механізми логічного розширення фізичної пам'яті. **У даній моделі реалізовано виділення сторінок процесам за запитами.** Призначені для користувача програми працюють з логічною пам'яттю, яка відображається на фізичну пам'ять посторінково. Фізична пам'ять розділена на сторінки однакового розміру, кожній сторінці може відповідати будь-яка сторінка логічної пам'яті. Фізично процес не обов'язково працює з безперервним блоком пам'яті, проте логічно сторінки пам'яті, з якими працює процес, залишаються суміжними. Тобто така форма заміщення не впливає на адресний простір програм. Логічні сторінки процесів, що не помістилися в основну пам'ять, знаходяться у вторинній пам'яті, що має, як правило, значно більший обсяг і меншу швидкодію. Спочатку в оперативну пам'ять завантажуються тільки перші сторінки процесів. Решта всіх сторінок завантажується згодом за запитами. У даній моделі використовується стратегія глобального витіснення, тобто будь-якій сторінці фізичної пам'яті може відповідати будь-яка сторінка будь-якого завдання.

Сторінковою відмовою називається ситуація, коли процес звертається до сторінки, що знаходиться у вторинній пам'яті. В цьому випадку виконується завантаження цієї сторінки в основну оперативну пам'ять. Якщо оперативна пам'ять заповнена, необхідно вивантажити одну зі сторінок у зовнішню пам'ять. Замінювана сторінка вибирається 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 знаходження в ОП було менше всього звернень.

NUR (Not Used Recently, давно не використовувана) – еталонна стратегія

заміщення, основана на прогнозі запитів процесів. Відбувається заміщення сторінки пам'яті, до якої найдовше не відбуватиметься звернень.

Менеджер пам'яті в даній моделі викликається при виникненні сторінкової відмови і проводить заміну сторінки згідно з вибраним алгоритмом. Тривалість **заміщення** однієї сторінки складає п'ять квантів модельного часу.

Файлова система

У ході роботи призначені для користувача процеси формують запит до ресурсів. Для взаємовиключення під час доступу до ресурсів у даній моделі файлова система використовує **двійкові семафори** (у реальних ОС семафорні операції виконуються безпосередньо ядром). Процес файлової системи запускається 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, очікуваних ресурсів).

Якщо процес звертається до якого-небудь ресурсу, виконується операція $P(S)$ над відповідним семафором, зменшуючи S на 1. Якщо при цьому значення $S=1$, процес отримує доступ до ресурсу, інакше – переводиться в стан очікування.

При звільненні процесом ресурсу виконується операція $V(S)$. Якщо при цьому є процес, що очікує звільнення ресурсу, то він дістає доступу до ресурсу, інакше семафор установлюється в 1.

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

Призначені для користувача процеси

У даній моделі може існувати до 10 призначених для користувача процесів.

Призначені для користувача процеси – деякі програми, що не мають відношення до системи, запускаються користувачем у своїх цілях (прикладні програми). Ці процеси мають найменший пріоритет.

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

– PC – указівник на операцію, що виконується процесом у даний момент;

– SP – указівник на вершину стека;

– DP – указівник 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 і ресурсів. У завдання ОС входить обслуговування цих запитів.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Робота з моделлю

Після запуску моделі на екрані з'являється заставка. Для початку роботи необхідно в головному меню вибрати пункт *Моделювання → Моделювання...* При цьому на екрані з'являється діалог параметрів моделювання.

Діалог параметрів моделювання

Діалог параметрів моделювання зображений на рис. 4.5. За допомогою діалогу можна виконувати такі дії:

1. Зміна параметрів функціонування моделі

– *Розмір кванта часу* – максимальний час, що відводиться процесу на використання CPU. Після закінчення кванта часу CPU віддається іншому процесу. Мі-

німальне значення 1. Значення за умовчанням 8. Максимальне значення 20.

– *Обсяг оперативної пам'яті* – вимірюється в сторінках. Мінімальне значення 1. Значення за умовчанням 10. Максимальне значення 20.

– *Стратегія заміщення сторінок* – алгоритм, використовуваний менеджером пам'яті для вибору сторінки, яка заміщається. Доступні значення: FIFO,LFU, NUR. Значення за умовчанням FIFO.

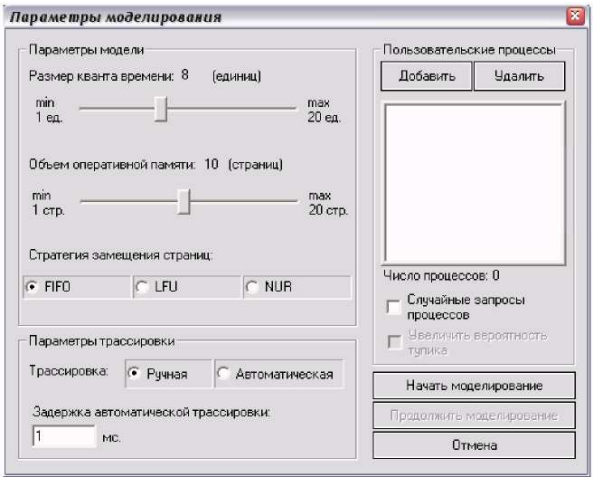


Рисунок 4.5 – Діалог параметрів моделювання

2. Зміна параметрів трасування:

– *Вид трасування.* При ручному трасуванні робота системи виконується покроково. Для просування модельного часу на одиницю вперед користувач повинен натиснути клавішу пробіл. При автоматичному трасуванні зупинка моделювання відбувається тільки при виникненні семафорної операції. Значення за умовчанням – ручне.

– *Затримка автоматичного трасування* – інтервал часу в мс, через який відбувається збільшення модельного часу при автоматичному трасуванні. Мінімальне значення 1. Значення за умовчанням 1. Максимальне значення 5000.

3. Призначені для користувача процеси:

– *Додавання процесу.* Для додавання призначеного для користувача процесу в систему необхідно натиснути кнопку «додати». Після натискання цієї кнопки з'являється діалог, що дозволяє задати ім'я процесу, який додається. За умовчанням процесам даються імена вигляду «Процес *X*», де *X* – його порядковий номер.

– *Видалення процесу.* Для видалення процесу необхідно виділити його в списку процесів і натиснути кнопку видалити.

– *Випадкові запити процесів.* Відмітка цього прапорця призводить до того, що процеси наново формуватимуть випадковим чином список запитів до системи під час кожного перезапуску моделювання. Зняття прапорця приводить до формування списку тільки один раз під час запуску програми, і процеси кожного разу формуватимуть одні й ті самі запити. Це дозволяє ставити чисті експерименти, не роблячи знижку на випадковість запитів процесів. За умовчанням прапорець знятий. Перевірити подібність запитів процесів можна, багато разів запускаючи моделювання і спостерігаючи стан системи у фіксований момент часу (стан системи має бути однаковим).

– *Збільшити вірогідність безвиході.* Відмітка цього прапорця призводить до формування списку запитів процесів таким чином, що з великою вірогідністю в системі виникне безвихідна ситуація. За умовчанням прапорець знятий.

4. Почати моделювання / Продовжити моделювання / Відміна:

– *Почати моделювання.* Натиснення цієї кнопки запускає процес моделювання з нульового моменту часу.

– *Продовжити моделювання.* Натиснення цієї кнопки спричиняє продовження моделювання з точки припинення. Таким чином, можлива динамічна зміна параметрів моделювання. У реальних системах параметри не змінюються динамічно, і дана можливість має чисто академічний характер.

– *Відміна.* Натиснення цієї кнопки закриває діалог без застосування змін.

Після натиснення кнопки «Почати моделювання» система переходить у режим моделювання (рис. 4.6).

Режим моделювання

Режим моделювання зображено на рис. 4.6.

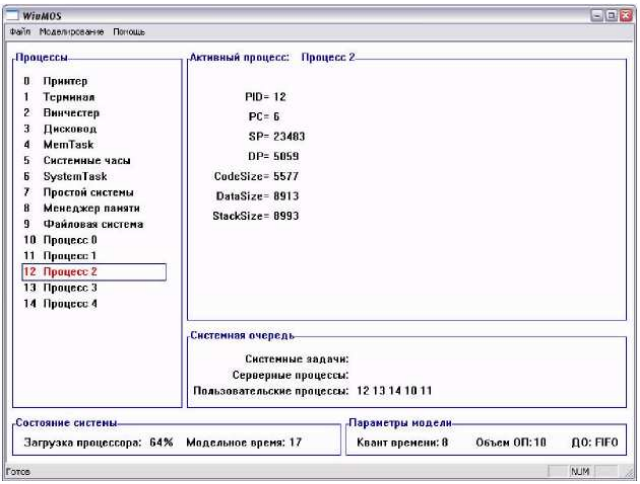


Рисунок 4.6 – Режим моделювання

Для зручності уявлення робоча область програми розбита на вікна, обмежені синіми прямокутниками. Розглянемо призначення і властивості кожного з них.

Вікно «Процеси». У цьому вікні відображається список процесів, що виконуються в системі. Процес, що виконується в даний момент на процесорі, виділений червоним кольором. Активний процес (процес, стан якого відображається у вікні «Активний процес») виділений темно-синьою рамкою. Процеси можна активізувати шляхом натиснення мишкою на їх назви в списку. Це дозволяє у будь-який момент бачити стан будь-якого процесу (по відношенню до призначених для користувача процесів – дескриптор).

Вікно «Активний процес». У цьому вікні відображається статус активного процесу (виділеного синьою рамкою у вікні «Процеси»). Статус включає 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 відображаються черги процесів, що очікують виділення їм 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», «Моделирование» і «Помощь». У підменю «Файл» міститься команда «Вихід» для виходу з системи. У підменю «Помощь» можна дізнатися інформацію про програму.

Підменю «Моделирование» 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 результуючим графіком.

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

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

Режим перегляду результатів

Режим перегляду результатів зображений на рис. 4.7.

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

По осі *X* відкладений модельний час.

По осі *Y* – процеси, що досі знаходяться в системі.

За допомогою кнопок «+» і «–» користувач може здійснювати масштабування графіка. За допомогою бігунка здійснюється навігація за графіком. При закритті діалогу програма повертається в режим моделювання.

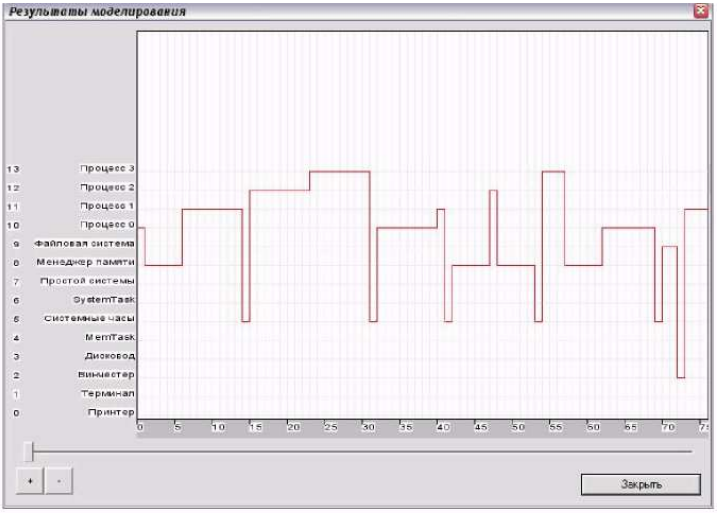


Рисунок 4.7 – Тимчасова діаграма режиму перегляду результатів

Дослідження результуючої тимчасової діаграми

Для оцінки точності результуючої тимчасової діаграми формується таблиця подій, які відбуваються в системі, і порівнюється вона з діаграмою. Для тестування візьмемо такі параметри системи: Число процесів користувача – 3. Обсяг оперативної пам'яті – 10 сторінок. Тривалість кванта – 5. Дисципліна обслуговування NUR (рис. 4.8).

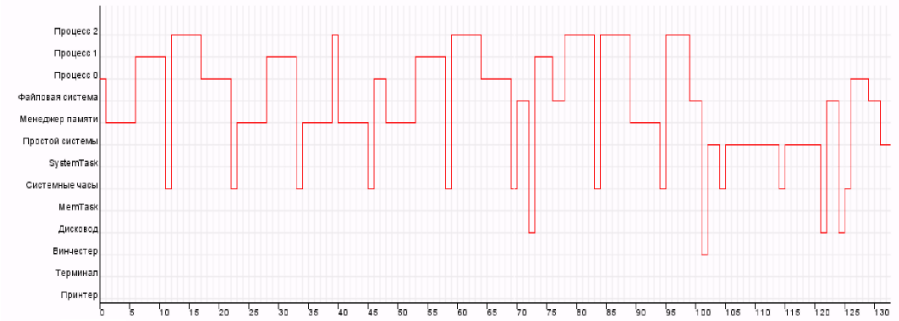


Рисунок 4.8 – Результуюча тимчасова діаграма

Подія:

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 Дисковод.

76 – Процес 11 звернувся до зайнятого ресурсу Дисковод.

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 «Дисковод» процесу 11.

125 – У черзі з’явився процес «Системний годинник».

129 – Процес 10 звернувся до зайнятого ресурсу «Вінчестер».

Оцінка якості моделі

Для оцінки якості моделі необхідно з’ясувати залежність вихідних параметрів від вхідних і зіставити отримані результати з теоретичними положеннями. Вхідними параметрами в даній моделі є: *число призначених для користувача процесів, обсяг оперативної пам’яті, стратегія заміщення сторінок і тривалість кванта часу*. Вихідні параметри: *кількість сторінкових відмов, завантаження*

процесора і *час завершення*. Отримаємо залежності вихідних параметрів від вхідних і спробуємо пояснити отримані результати.

Число сторінкових відмов. Залежність числа сторінкових відмов від кількості сторінок ОП і алгоритму заміщення сторінок.

Для оцінки цієї залежності побудуємо в одних осях графіки залежності числа сторінкових відмов від числа сторінок при використанні різних дисциплін обслуговування (ДО). Для об'єктивності виконаємо тестування при різних значеннях числа призначених для користувача процесів (табл. 4.1, a, b, c).

Таблиця 4.1, a – Результати тестування кількості сторінкових відмов (квант часу = 8,число процесів = 2)

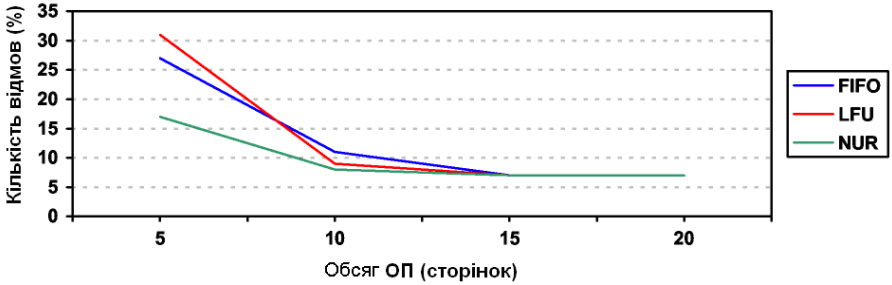
Число сторінок	Відсоток відмов		
	FIFO	LFU	NUR
5	27	31	17
10	11	9	8
15	7	7	7
20	7	7	7

Таблиця 4.1, b – Результати тестування кількості сторінкових відмов (квант часу = 8, число процесів = 6)

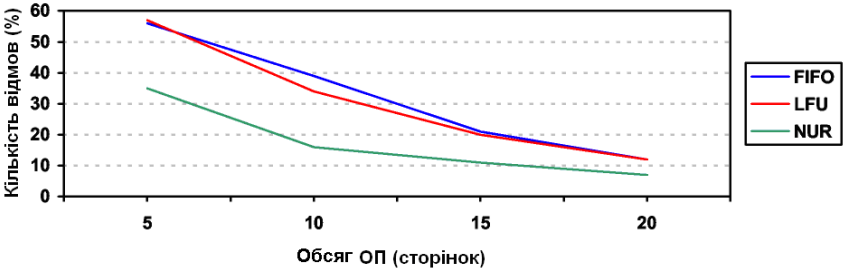
Число сторінок	Відсоток відмов		
	FIFO	LFU	NUR
5	56	57	35
10	39	34	16
15	21	20	11
20	12	12	7

Таблиця 4.1, c – Результати тестування кількості сторінкових відмов (квант часу = 8, число процесів =10)

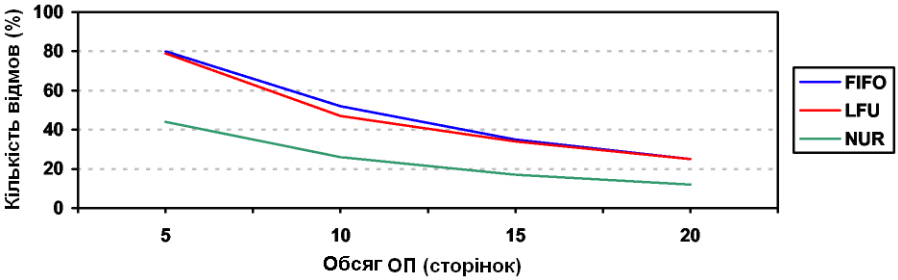
Число сторінок	Відсоток відмов		
	FIFO	LFU	NUR
5	80	79	44
10	52	47	26
15	35	34	17
20	25	25	12



Параметри тестування: квант часу = 8, число процесів = 2



Параметри тестування: квант часу = 8, число процесів = 6



Параметри тестування: квант часу = 8, число процесів = 10

Рисунок 4.9 – Залежність кількості сторінкових відмов від обсягу оперативної пам’яті

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

– зі збільшенням ОП число 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 кращі результати, чим FIFO. Дисципліна NUR дає майже в два рази менше сторінкових відмов, чим FIFO і LFU. Ця ДО зоснована на передбаченні запитів процесів і є еталонною;

– на рис. 4.9 при числі сторінок 15 і 20 усі три ДО дають однакові результати. Це означає, що системі вдалось розмістити всі процеси ОП і операцій заміщення сторінок не відбувається. Передумови до виникнення такої ситуації дає великий обсяг ОП і мале число процесів;

– зі збільшенням числа процесів зростає кількість сторінкових відмов. Це результат дефіциту ресурсу пам’яті для великої кількості процесів.

Залежність числа сторінкових відмов від кванта процесорного часу

Імовірно число сторінкових відмов не залежить від кванта часу, проте цю гіпотезу необхідно перевірити. Була оцінена залежність кількості сторінкових відмов від параметра моделювання кванту часу при інсталяції системи 5 призначених для користувача процесів (табл. 4.2).

Таблиця 4.2 – Результати тестування залежності кількості відмов від кванта часу (обсяг ОП = 10, ДО – LFU)

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

Виявилось слабке зменшення кількості сторінкових відмов при великих значеннях кванта часу (рис. 4.10).

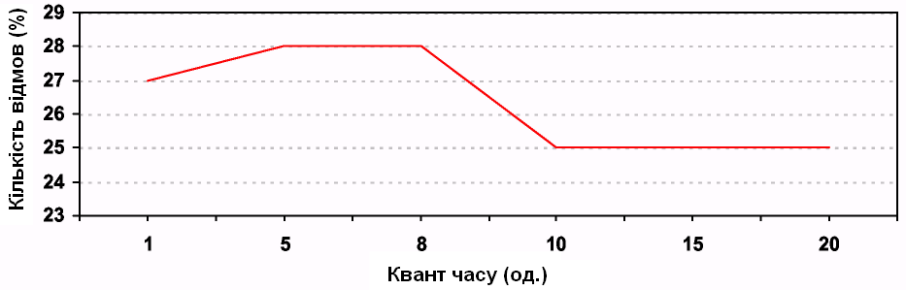


Рисунок 4.10 – Залежність кількості відмов від кванта часу

Збільшення часу безперервного знаходження кожного процесу на процесорі, що супроводжується зверненнями до одних і тих самих сторінок пам'яті, і, як наслідок, деяке зменшення хаотичності запитів до сторінок пам'яті приводить до зменшення числа сторінкових відмов.

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

Оцінка залежності завантаження процесора від алгоритму заміщення сторінок.

Для оцінки цієї залежності побудуємо в одних осях графіки залежності завантаження процесора від часу при використанні різних ДО, на яких передбачається зафіксувати три інтервали:

1 Інтервал наростання завантаження. В цей час призначені для користувача процеси надаються системі і 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 спаду. Спад відбувається, коли запити процеси користувача вже оброблені, і процеси не потребують роботи на процесорі. Чим раніше настає цей етап, тим краще працює система.

Для об’єктивності виконаємо тестування за різних значень числа призначених для користувача процесів (табл. 4.3, a, b).

Таблиця 4.3, a – Результати тестування завантаження процесора в часі (квант часу = 8, число процесів = 8, Обсяг ОП = 10)

ТМ	Завантаження CPU		
	FIFO	LFU	NUR
50	16	16	16
100	36	36	42
200	39	39	50
400	45	46	51
600	45	45	54
800	47	46	54
1000	46	46	49
1200	45	45	50
1400	46	45	52
1600	47	47	48
1800	43	43	43
2000	38	38	38
2200	35	35	35
2400	32	32	32
2600	29	29	28

Таблиця 4.3, b – Результати тестування завантаження процесора в часі (квант часу = 8, число процесів = 3, Обсяг ОП = 10)

ТМ	Завантаження CPU		
	FIFO	LFU	NUR
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
1000	42	42	42
1200	35	35	35
1400	30	30	30
1600	26	26	26

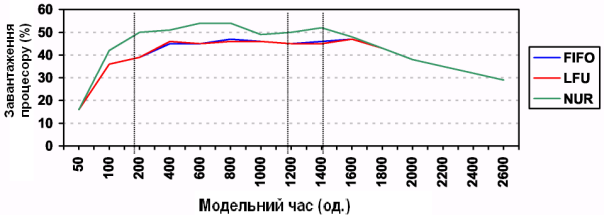
На рис. 4.11 зображено графіки залежностей CPU від модельного часу, на основі яких можна зробити такі висновки:

– достатньо точно простежуються інтервали зростання, максимуму і спаду;

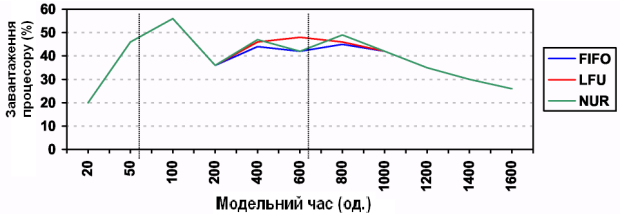
– для восьми процесів (рис. 4.11, *a*) дисципліна NUR дає більше завантаження процесора, чим LFU і FIFO. Це пов'язано з меншим числом сторінкових відмов, що даються ДО = NUR. Для трьох призначених для користувача процесів (рис. 4.11, *b*) різні ДО дають схожі результати. Це пояснюється тим, що системі вдалося цілком, або майже цілком, розмістити процеси оперативної пам'яті (у зв'язку з малою їх кількістю), і число сторінкових відмов мале незалежно від алгоритму заміни сторінок;

– для восьми процесів інтервал спаду для дисципліни NUR почався раніше. Системі вдалося швидше задовольнити запити призначених для користувача процесів, і це ще раз підтверджує високу ефективність дисципліни NUR;

– дисципліни LFU і FIFO дали схожі результати. Отже, деяка перевага LFU перед FIFO не впливає значно на завантаження процесора.



a) Параметри тестування: квант часу = 8, число процесів = 8, Обсяг ОП = 10



b) Параметри тестування: квант часу = 8, число процесів = 3, Обсяг ОП = 10

Рисунок 4.12 – Залежність завантаження CPU від модельного часу

Оцінка залежності завантаження процесора від величини кванта часу

Для оцінки цієї залежності побудуємо в одних осях графіки залежності завантаження процесора від часу при різних значеннях кванта часу (рис. 4.12). Конфігурація системи при тестуванні: 5 призначених для користувача процесів, ДО = FIFO, обсяг 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
1000	46	46	49	46
1200	46	49	51	48
1400	47	48	48	48
1600	46	47	47	46
1800	42	42	42	42
2000	38	38	38	38
Середнє завантаження	40,92	42,00	43,31	41,38

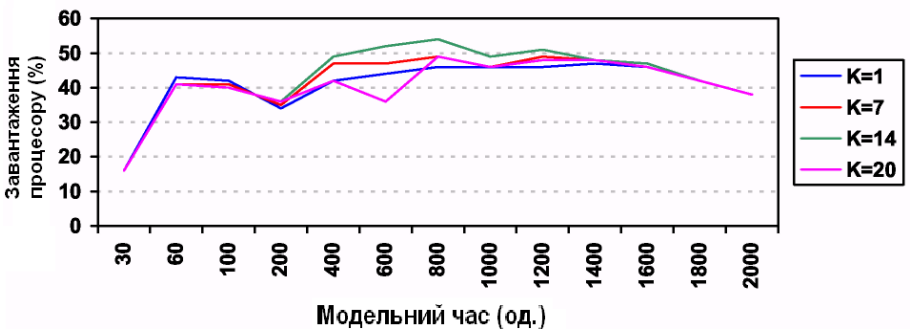


Рисунок 4.13 – Залежність завантаження CPU від модельного часу

На графіку (див. рис. 4.12) видно області наростання, максимуму і спаду завантаження процесора, проте залежність завантаження процесора від кванта часу не наочна.

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

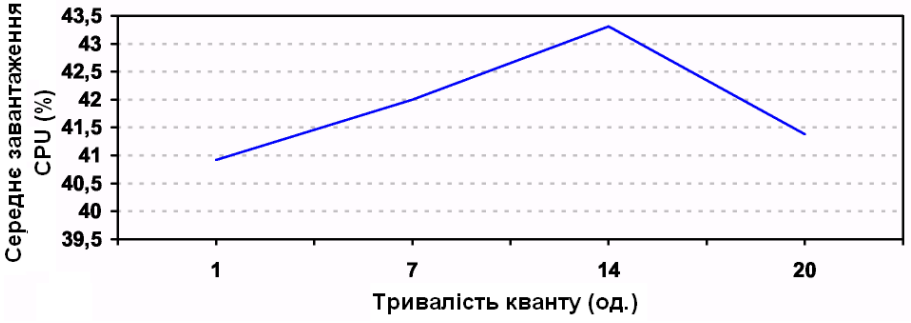


Рисунок 4.14 – Залежність завантаження CPU від тривалості кванта часу

З графіка можна зробити такі висновки:

– при невеликій тривалості кванта з її зростанням зростає і завантаження процесора. Це пояснюється тим, що чим більше час безперервного використання процесом CPU, тим більше число запитів може сформувати процес і в результаті з більшою ймовірністю не чекатиме в системній черзі чергового кванта процесорного часу, а займе який-небудь ресурс. Крім того, зі збільшенням кванта часу система рідше виконуватиме власні завдання, наприклад, системний годинник;

– при великій тривалості кванта завантаження процесора падає. Це пов’язано з тим, що система менш оперативно відповідає на запити процесів.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Розробити модель функціонування операційної системи при обробці

системних і призначених для користувача процесів і провести дослідження її параметрів за заданим варіантом (табл. 4.5).

Таблиця 4.5 – параметри операційної системи

Вар.	Розмір кванта	Обсяг оперативної пам'яті	Кількість користувачевих процесів	Стратегія заміщення (К)	Тривалість моделювання
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
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 за журналом групи.

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

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

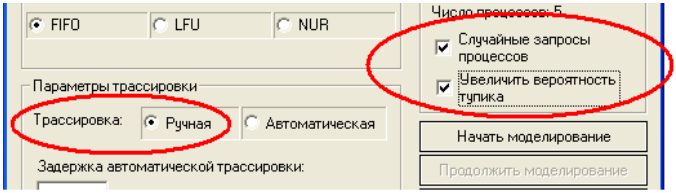


Рисунок 4.14 – Обов'язкові параметри моделі

Імена призначених для користувача процесів задаються у форматі: *XX_YY_ZZ_#*, де *XX* – перші дві букви прізвища студента; *YY* – ініціали студента; *ZZ* – дві цифри місяця народження студента *#* – порядковий номер самого процесу. При цьому **всі імена процесу задаються заголовними символами англійського алфавіту.**

Задані за варіантом параметри і початковий стан моделі перед стартом (початком моделювання) подаються у звіті.

У процесі моделювання необхідно отримати і представити в звіті (у вигляді графіків, таблиць) таку інформацію:

1. Кількість викликів системного годинника, шт.
2. Кількість викликів завдань, обслуговуючих ресурси (Принтер, Термінал, Вінчестер, Дисковод), – про кожний ресурс окремо.
3. Кількість викликів System Task, шт.
4. Кількість простоїв і їх загальна тривалість у квантах.
5. Кількість сторінкових відмов.
6. Кількість операцій *P*(*S*) і *V*(*S*), семафор якого пристрою при цьому спрацював, хто був власником ресурсу, який процес чекав і який процес звільнив ресурс у вигляді таблиці 4.6.

Таблиця 4.6 – Результуюча таблиця

№ п/п	Квант часу	P(S)	V(S)	Семафор	Власник, PID	Черга процесів			Звільняючий процес, PID
Початковий стан									
				Вінчестер	-1				
				Дисковод	-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 і так далі необхідно зареєструвати всі параметри призначених для користувача процесів, активних у даний момент, як показано на рис. 4.15 у вигляді таблиці 4.7.

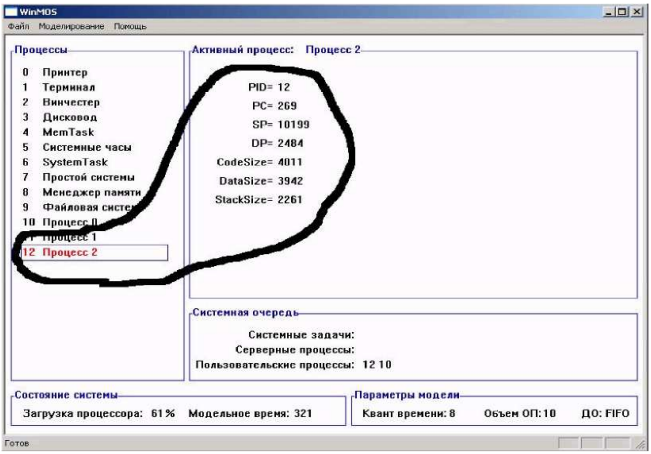


Рисунок 4.15 – Параметри активного призначеного для користувача процесу

Таблиця 4.7 – Результуюча таблиця

№ п/п	Квант часу	PID	PC	SP	DP	CodeSize	DataSize	StackSize
1	1	11	269	199	243	4011	3942	2261
2	7	12	290	200	243	4016	3972	5261

8. Результати моделювання (**графік залежності роботи процесора від модельного часу з програмою з його описом**) без масштабування – склеєний за всією довжиною тривалості експерименту.

9. Залежності завантаження процесора від алгоритму заміщення сторінок (у вигляді **таблиць і відповідних графіків**), як показано в прикладах на рис. 4.10 та таблиці 4.3. Дані при цьому обробляти для кількості квантів часу ***X* + 50**, де *X* – задана за варіантом кількість квантів часу; обсяг ОП – фіксований. Висновки за даними залежностей.

10. Залежність числа сторінкових відмов від числа сторінок ОП і алгоритму заміщення сторінок (у вигляді **таблиць і відповідних графіків**), як показано на прикладах рис. 4.8 та таблиці 4.1. Дані при цьому 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 даними залежності.

11. Залежність числа сторінкових відмов від квантів процесорного часу (у вигляді таблиці і відповідного графіка), як показано в прикладах рис. 4.9 та таблиці 4.2. Надати висновки за даними залежності.

12. Залежність завантаження процесора від величини кванта часу (у вигляді таблиць і відповідних графіків), як показано в прикладах рис. 4.11 та таблиці 4.4. Дані при цьому обробляти для кількості квантів часу з кроком 50. Надати висновки за даними залежності.

13. Графік залежності середнього завантаження процесора від тривалості кванта на основі прикладу рис. 4.12 і даних, отриманих у п.12. Надати виснов-ки за даними залежності.

Зміст звіту

Звіт повинен містити:

1. назву, дату виконання, тему, мету, опис завдання, матеріальне забезпечення, хід виконання і результати лабораторної роботи;

2. висновки для пунктів 1–13 і загальні висновки по роботі.

Контрольні запитання

1. Яким чином взаємодіють процеси при моделюванні?
2. Як взаємодіють резидентна і нерезидентна частина таблиці управління процесами?
3. Яким чином здійснюється синхронізація процесів?
4. Поясніть особливості планування процесів?
5. Поясніть особливості алгоритмів заміщення сторінок та вимоги до них?
6. Яка роль в лабораторній роботі належать семафору? Назвіть його можливі значення.

Лабораторна робота 5

ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ ПРОЦЕСІВ І ПОТОКІВ

Мета роботи: ознайомитись з концепцією мультипрограмування і багатопотоковості в сучасних ОС, набути практичних навичок з тестування, аналізу зареєстрованих у них процесів і потоків, виділення специфічних властивостей і характеристик процесів за рахунок використання різних програмних продуктів, проведення аналітичного порівняння наведених характеристик.

Указівки з підготовки до виконання лабораторної роботи

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

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

Додаткову інформацію можна отримати у таких джерелах:

1. Таненбаум Э. Современные операционные системы / Э. Таненбаум – СПб.: Питер, 2010. – С. 112 – 215.

2. Шеховцов В.А. Операційні системи. – Київ .: Видавнича група BHV, 2005. – С. 45 – 109.

Теоретичні відомості

В операційній системі Windows підтримуються багатопотокові процеси. ***Процес (process)*** – це об’єкт, якому належать ресурси додатку. ***Потік (thread)*** – це суть усередині процесу, яку ядро Windows направляє на виконання. Без нього

програма процесу не може виконуватись. Потік розділяє разом з процесом загальний адресний простір, код і глобальні дані. У кожного потоку є власні регістри, стік і механізми введення, у тому числі 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 перевантажений великою кількістю програм, то користувачеві здасться, що всі ці програми виконуються паралельно.

Багатопотоковість – це можливість програми бути багатозадачною. Програма може бути розділена на окремі потоки виконання, які виконуються паралельно.

Для дослідження процесів і потоків існують дві основні групи програмних засобів: вбудовані в операційну систему і сторонніх виробників.

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

До програм сторонніх виробників, розгляд та дослідження за їх допомогою присвячена ця лабораторна робота, 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 інші.

Диспетчер завдань Windows. У диспетчері завдань відображаються відомості про програми і процеси, що виконуються на комп'ютері (рис. 5.1). Крім того, можна проглянути найбільш часто використовувані показники швидкодії процесів.

Диспетчер завдань служить для відображення ключових показників швидкодії комп'ютера. Для виконуваних програм можна проглянути їх стан і завершити програми, що припинили відповідати на запити. Є можливість перегляду акти-

вності процесів, що виконуються, з використанням до 15 параметрів, а також графіків і відомостей про використання центрального процесора (ЦП) і пам'яті.

Якщо є підключення до мережі, можна переглянути стан мережі і параметри її роботи. Якщо до комп'ютера підключено декілька користувачів, можна побачити їх імена, які завдання вони виконують й відправити їм повідомлення.

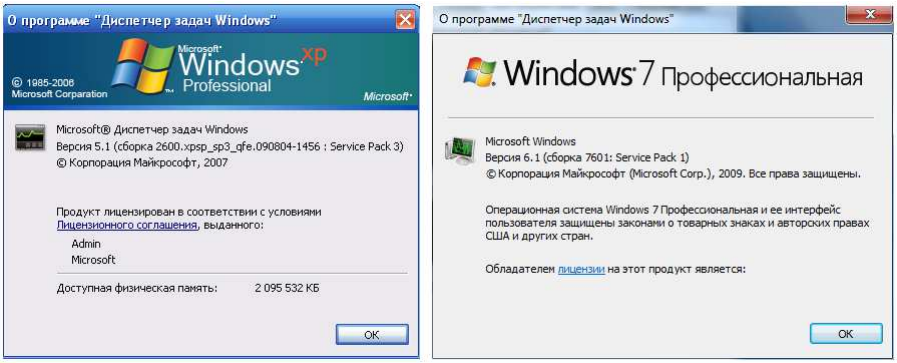


Рисунок 5.1 – Диспетчер завдань Windows

Для керування процесами Диспетчер завдань Windows містить вкладку «Процеси». У вкладці є можливість відстежувати виконання процесів за допомогою таких лічильників, що відображаються в заголовках стовпців:

1. Ім'я образу – ім'я процесу в диспетчерові завдань.

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

3. Базовий пріоритет – ранг, що визначає порядок, в якому потоки процесу обробляються процесором. Зміна пріоритету виконується за допомогою пункту "Пріоритет" контекстного меню. Можливі значення наведені у таблиці 5.1.

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

5. Завантаження ЦП – процентна частка часу, протягом якого процес використовував ЦП з моменту останнього оновлення.

6. Обсяг віртуальної пам’яті – розмір адресного простору, переданого процесу.

7. Пам’ять – максимум – обсяг фізичної пам’яті, використаної процесом з моменту його запуску.

8. Пам’ять – використання – поточний набір сторінок пам’яті, зайнятих процесом (у Кбайт). Поточний набір – це число сторінок, резидентних зараз у пам’яті.

Таблиця 5.1 – Класи пріоритетів процесу

№ п/п	Клас пріоритету	Значення	Опис
1.	Реального часу	24	Процес з таким пріоритетом витісняє навіть дії ОС. Цей клас слід застосовувати тільки у разі абсолютної необхідності
2.	Високий	13	Процес з високим пріоритетом. Цей прапор надає вплив на можливість виконання інших процесів
3.	Вище середнього	10	Цей прапор позначає процес, пріоритет якого перевищує середній клас, проте нижче за високий клас
4.	Середній	8	Основний пріоритет процесу (задано за умовчанням)
5.	Нижче середнього	6	Цей прапор позначає процес, пріоритет якого перевищує низький клас, проте нижче середнього класу
6.	Низький	4	Процес з найнижчим пріоритетом. Виконується в період простою системи

9. Пам’ять – зміна – кількість пам’яті (у Кбайт), використаної з моменту останнього оновлення процесом.

10. Об’єкт USER – об’єкт диспетчера вікон, що включає вікна, меню, курсори, значки, оброблювачі, поєднання клавіш, і інші внутрішні об’єкти.

11. Об’єкт GDI – об’єкти бібліотеки GDI інтерфейсів програмування для графічних пристроїв.

12. Лічильник потоків – число потоків, що виконуються процесом.

13. Лічильник дескрипторів – кількість дескрипторів об’єктів у таблиці об’єктів процесу в диспетчері завдань.

14. Вивантажуваний пул (куча) – віртуальна пам’ять, виділена системою процесу для розміщення програмного коду і даних, які можуть бути вивантажені з оперативної пам’яті. Підкачка сторінок – це переміщення рідко використовуваних

частин коду програми з оперативної пам'яті на інший (системний) носій. Зазвичай використовується жорсткий диск.

15. Невигружений пул – пам'ять, зайнята операційною системою, ніколи не вивантажувана на диск й інші носії.

Єдиний з лічильників, яким може керувати користувач – базовий пріоритет процесу.

Aida. Безкоштовна професійна програма, яка випускається у двох версіях: ***Aida32*** і ***Aida64.***

Aida32 – професійний інструмент для діагностики обладнання і аналізу системної конфігурації (рис. 5.2). Аналізує комп'ютер і надає детальну інформацію як про апаратну частину (процесор, материнська плата, монітор і відеопідсистема цілком, диски і так далі), так і про програмну складову – ОС, драйвери, всі встановлені і окремо автозавантажувані програми, запущені процеси, ліцензії і т.д.

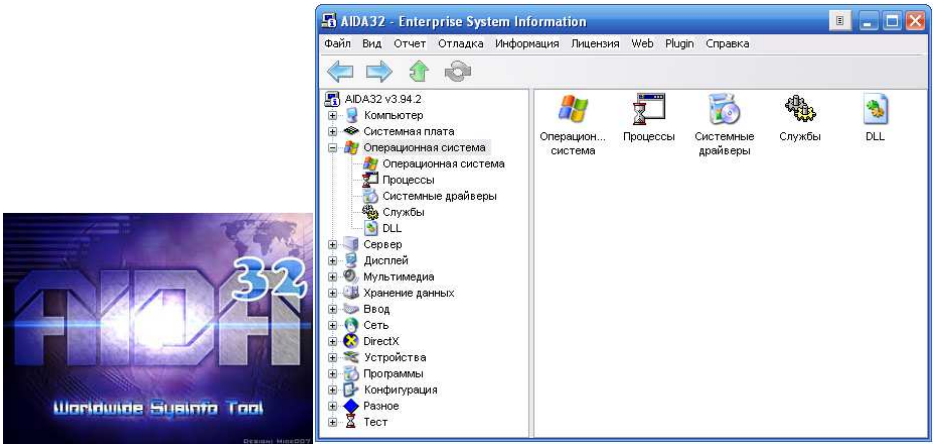


Рисунок 5.2 – Програмний продукт Aida32

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 віддалених комп'ютерів по мережі TCP/IP. Зокрема показує характеристики:

1) апаратна конфігурація: центральний процесор, материнська плата, монітор, відеоплата, встановлені жорсткі диски, BIOS і так далі;

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

Програма тестує продуктивність комп'ютера і звіряє їх з еталонними результатами для відображення загальної картини швидкості вашого комп'ютера.

Aida64 (колишній EVEREST) – утиліта є могутнім засобом для ідентифікації і тестування практично будь-яких компонентів персонального комп'ютера під керуванням ОС сімейства Windows (рис. 5.3). В процесі оптимізації та тонкого налаштування програма забезпечує отримання необхідної системної інформації, надає просунуті можливості моніторингу і діагностики апаратного забезпечення для оцінки ефекту, досягнутого застосуванням тих або інших налаштувань. Тести продуктивності центрального процесора, модуля обчислень з плаваючою точкою, а також пам'яті – допомагають визначати реальну продуктивність системи і порівняти її з раніше отриманими результатами або з іншими комп'ютерами.

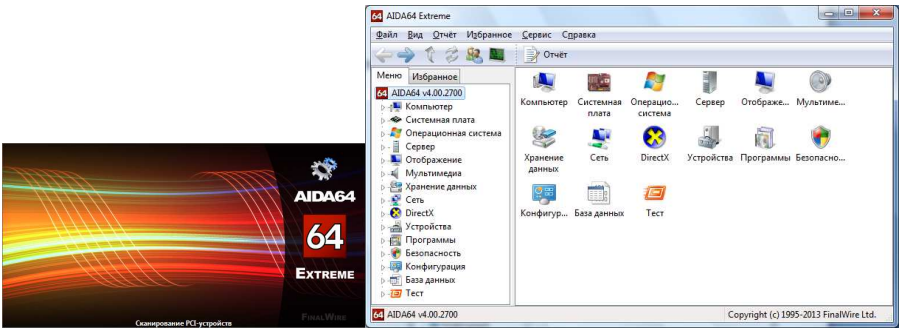


Рисунок 5.3 – Програмний продукт Aida64

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

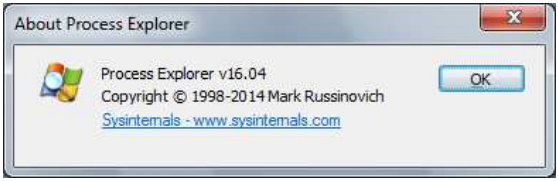


Рисунок 5.4 – Програмний продукт Process Explorer

Process Explorer має могутню систему пошуку, що дозволяє шукати процеси, що відкривають специфічний дескриптор або які завантажують певні динамічні бібліотеки DLL.

За допомогою Process Explorer можна зберегти в текстовому файлі список усіх процесів з описами і розміром зайнятої кожним з них пам'яті, запустити будь-який додаток, вимкнути, перезавантажити або заблокувати комп'ютер, знайти використовувані бібліотеки, включити підсвічування кольором певних процесів і т.д. Все це допомагає контролювати працюючі процеси і 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 ікону і компанію виробника кожного процесу.
- змінний діапазон вимірювань завантаження CPU і графічні індикатори;
- можливість «заморозити» будь-який процес;
- можливість керування (запуск, пауза, зупинка) потоками процесу;
- можливість вивести вікно, що належить тому або іншому процесу поверх останніх;
- можливість закриття дерева процесів;
- можливість у реальному режимі часу міняти пріоритет і те, на якому ядрі процесора виконуватиметься той або інший процес;
- можливість перевірки сертифіката файлу процесу;

− можливість замінювати системний диспетчер завдань за тими ж гарячими клавішами;

− для кожного об’єкта, що має ACL, відображається вкладка «Безпека».

Утиліта не вимагає інсталяції. Досить запустити файл ***procexp.exe***, відкриється головне вікно утиліти (рис. 5.5).

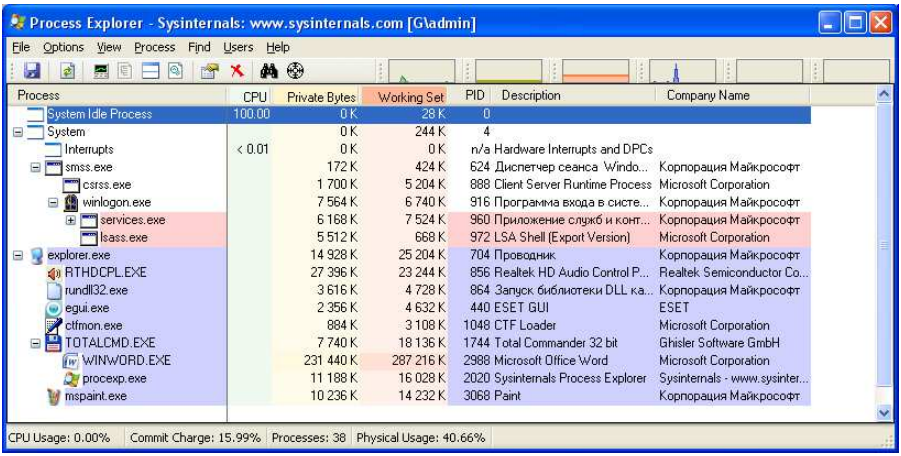


Рисунок 5.5 – Головне вікно утиліти Process Explorer

У верхній частині вікна в деревоподібній структурі перераховані всі процеси, що працюють у системі. Окрім імені процесу, виводиться інформація про використання процесора, опис процесу, обсяг зайнятої пам’яті. Подвійне клацання по імені процесу відкриває вікно його властивостей (рис. 5.6).

У нижній частині головного вікна надана детальна інформація про виділений у верхній частині процес. Це може бути список бібліотек, використовуваних процесом, або список дескрипторів (відкритих файлів, ключів реєстру і т.д.). Подвійний клік миші на ім’я бібліотек або дескриптора відкриває вікно властивостей. Для отримання опису бібліотеки за допомогою кліку правої кнопки миші по її імені можна відкрити меню, в якому пункт «Google» відправляє запит у пошукову систему з ім’ям вибраної бібліотеки.

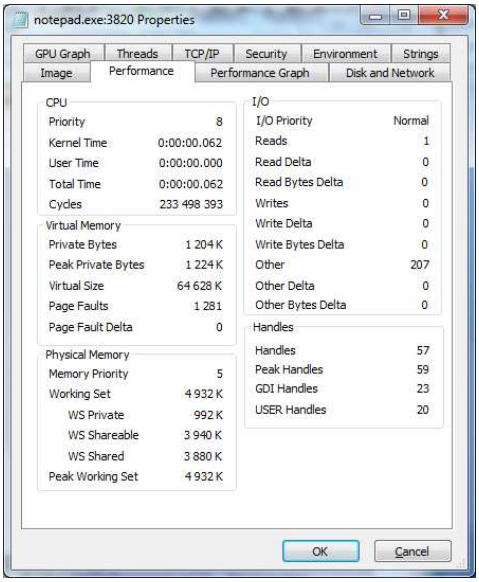


Рисунок 5.6 – Вікно властивостей процесу

Для будь-якого процесу за допомогою клацання правої кнопки миші по імені процесу можна змінити його пріоритет, завершити його виконання або завершити виконання процесу і його дерева. Корисна особливість Process Explorer полягає в тому, що роботу будь-якого процесу можна припинити (*suspend*), а потім відновити (*resume*). Припинення роботи процесу можуть тимчасово звільнити зайняті ним ресурси (процесор, мережа, жорсткий диск) для використання іншими прикладними програмами. Припинивши виконання процесу, можна визначити, наприклад, який процес відкриває вікна або проявляє підвищену мережеву активність. Це необхідно зробити при підозрі 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 підсвічуватиметься ім'я шуканого процесу.

Process Explorer може вивести приховане вікно додатку на екран. Деякі програми не відновлюють свій значок в tray, і якщо оболонка перезавантажується, то після цього доступ до інтерфейсу, згорнутого в tray програми, отримати буде неможливо. Доведеться завершувати програму за допомогою Диспетчера завдань і запускати наново з втратою всіх незбережених даних. Скориставшись пунктом контекстного меню *Bring to Front*, можна дістати доступ до вікна і зберегти дані.

Крім того, за допомогою Process Explorer можна вивантажити до текстового файлу список усіх процесів з описами і об'ємом пам'яті зайнятим кожним з них, запустити будь-який додаток, вимкнути, перезавантажити або заблокувати комп'ютер, знайти використовувані бібліотеки та дескриптори, включити виділення кольором певних процесів та інше. Усі дії допомагають контролювати працюючі процеси і отримувати інформацію про використання системних ресурсів.

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

Рисунок 5.7 – Утиліта Process Monitor

Програма поєднує в собі такі можливості (рис. 5.8):

- відстеження запуску і завершення роботи процесів і потоків, включаючи інформацію про код завершення;
- відстеження завантаження образів (бібліотек DLL і драйверів пристроїв, що працюють у режимі ядра);
- збір даних про параметри операцій введення і виведення;

– наявність нешкідливих фільтрів, що дозволяють встановлювати фільтри, які не призводитимуть до втрати даних;

– збирання стеків потоків для кожної операції дозволяє в більшості випадків визначити початкову причину виконання операції;

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

– налаштування та переміщення колонок для кожної властивості події;

– встановлення фільтрів на будь-яке поле з даними, включаючи поля, які не є колонками;

– розширена можливість програми до десятків мільйонів зареєстрованих подій та гігабайтів даних про події – вдосконалена архітектура запису журналів;

– відображення стосунків між усіма процесами, перерахованими у відомостях трасування за допомогою дерева процесів;

– збереження всіх даних у журналі, щоб їх можна було завантажити в іншому екземплярі програми Process Monitor;

– підказки для простого перегляду інформації про образ процесу;

– детальні підказки дозволяють отримати зручний доступ до форматованих даних, які не вміщуються в колонці;

– пошук, що припиняється;

– запис у журнал усіх операцій під час завантаження системи.

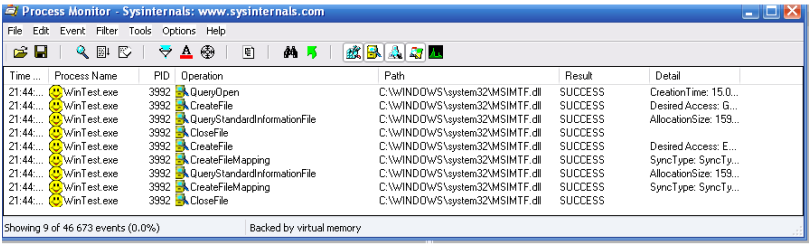


Рисунок 5.8 – робоче вікно програми Process Monitor

Ознайомитись з можливостями програми Process Monitor можна прочитавши файл довідки і спробувавши скористатись кожним пунктом меню та налаштуванням програми на робочій системі.

Щодо аналізу процесів Process Monitor надає таку унікальну можливість, як показ сумарної динамічної інформації в графічному вигляді за весь період існування процесу. Наприклад, для процесу WinTest.exe зображено на рис. 5.9.

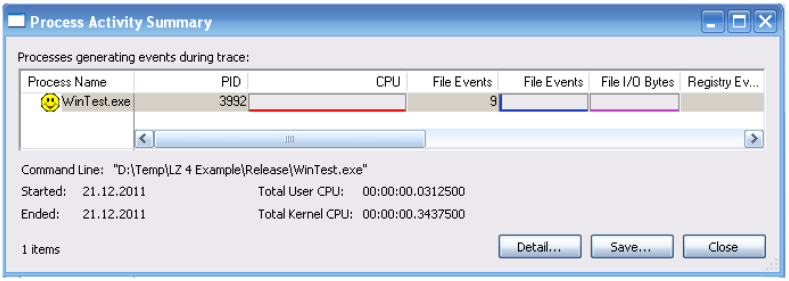


Рисунок 5.9 – Аналіз WinTest.exe процесу

Деталізація даних дозволяє оцінити ступінь впливу на ОС (рис. 5.10).

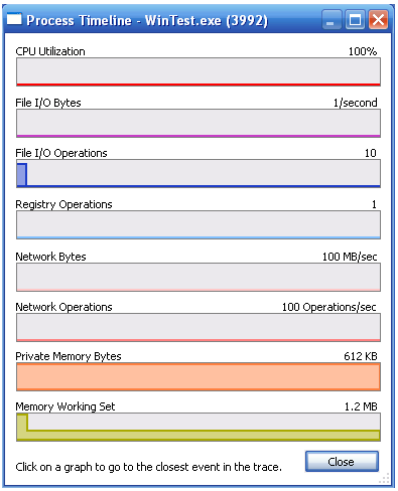


Рисунок 5.10 – Деталізація даних аналізу WinTest.exe процесу

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

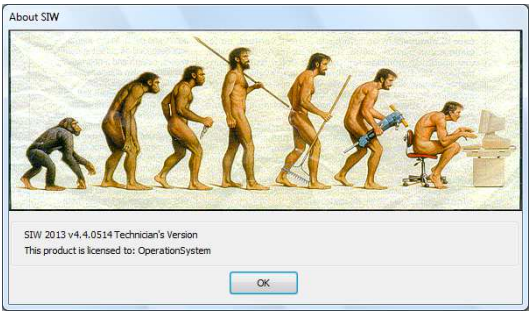


Рисунок 5.11 – Програмний продукт System Info for Windows

SIW відображає інформацію про ОС, оновлення, системні папки та файли, встановлені програми, запущені процеси в детальному вигляді, надаючи список усіх використовуваних файлів вибраного процесу, драйверів та служб, установленого обладнання, включаючи інформацію про материнську плату, процесор, BIOS і т.д. За отриманими даними можна створити файл звіту у форматах CSV, HTML, TXT або XML. Інтерфейс програми перекладений на багато мов. Інтерфейс програми розділений на 3 категорії: «Обладнання», «Програми», «Мережа» (рис. 5.12).

Категорія «Обладнання». SIW надає користувачам детальну інформацію про кожен елемент обладнання, зокрема процесор, PCI, мережеві адаптери, материнську плату, BIOS, пам'ять, відеопристрої, принтери, сенсори, оптичні і жорсткі диски, системні слоти, електроживлення, звукові пристрої, ресурси і т.д.

Категорія «Мережа». Утиліта здатна провести широкомасштабний аналіз мережі та відобразити детальні відомості про неї, зокрема про відкриті порти, ме-

режеве оточення, загальний доступ до файлів, видалених підключеннях і т.д.

Категорія «Програми». SIW проводить розширений пошук з подальшим виведенням інформації на дисплей про наявність встановленого програмного забезпечення і компонентів у системі, зокрема про операційну систему, встановлені оновлення та програми, завантажені та розділені DLL, системні папки та файли, драйвери, ліцензійні ключи до програм, автозавантаження, збереження паролів на web-серверах, спеціальні можливості, змінні середовища, бази даних, регіональні налаштування, типи файлів, відкриті файли, запущені процеси, захищені файли, ActiveX і т.д.

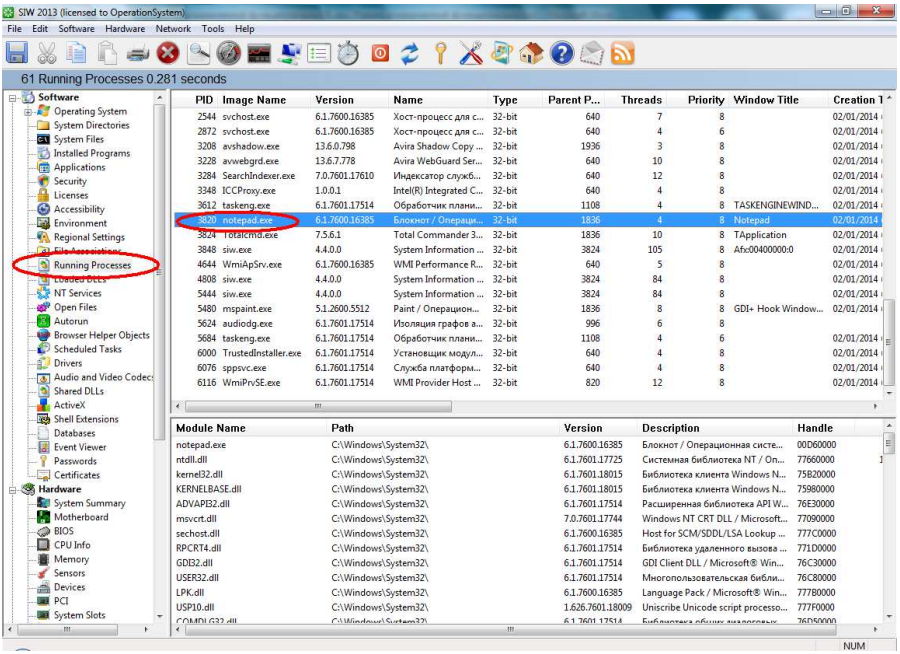


Рисунок 5.12 – Інтерфейс System Info for Windows

TaskInfo – це програма для повного моніторингу у реальному часі всіх системних процесів. Дозволяє відстежувати не тільки використання пам’яті та CPU, але й відкриті файли, використовувані бібліотеки і т.д. (рис. 5.13).

Рисунок 5.13 – Про програму TaskInfo

TaskInfo дозволяє змінювати пріоритет будь-якого процесу, зупиняти його або запускати. Результати моніторингу можна відстежувати візуально як у графічному режимі, так і в текстовому (рис. 5.14).

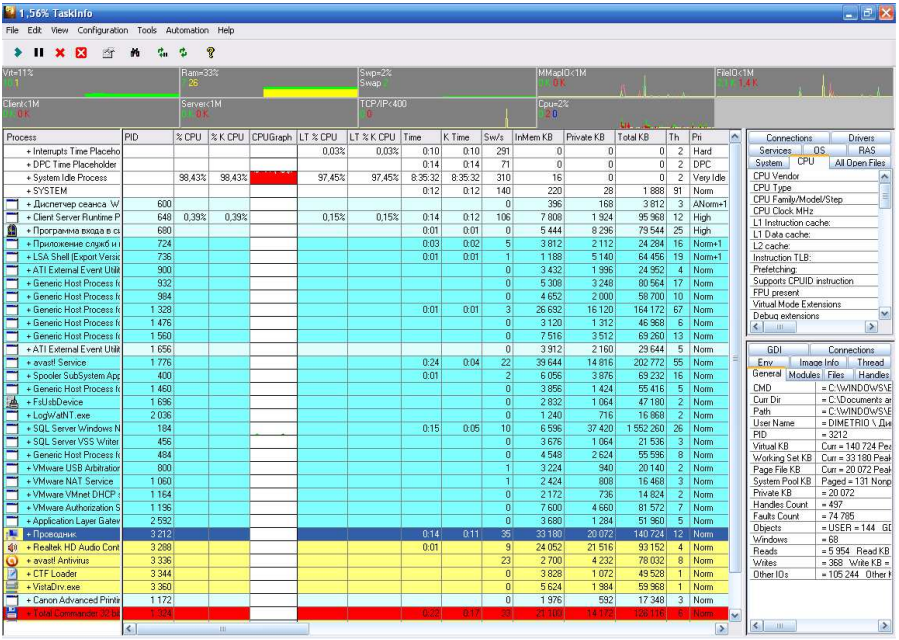


Рисунок 5.14 – Вікно результатів моніторингу

TaskInfo може замінити зв'язку зі стандартним *Диспетчером завдань* та інструментом системної інформації. TaskInfo дозволяє отримати такі дані:

- список усіх запущених процесів і потоків (зокрема системних потоків), з детальною інформацією про кожен процес: використання процесора і пам'яті, шлях, усі відкриті файли та модулі (DLL), параметри командного рядка, змінні оточення, відкриті з'єднання та інше;
- список більшості процесів, які намагаються бути невидимими, наприклад, черв'яки, key logs й інше шпигунське програмне забезпечення;
- використання процесора, пам'яті (фізичної, віртуальної, кешу і swap);
- детальна інформація про встановлені процесори і операційну систему;
- інформація про дані на локальних дисках, мережевих і т.д.;
- усі відкриті файли, драйвери і TCP/IP, деталі VPN-з'єднання.

Функціональність програми TaskInfo: копіює усю інформацію до буфера обміну або текстового файлу; запуск/зупинка процесів та вимкнення/перезапуск системи; можливість використання з командного рядка; автоматично показує сповіщення про різні умови обмеження ресурсів; звільнення фізичної пам'яті за вимогою; прискорене завершення некоректно працюючих процесів; зміна пріоритету процесу; перезапуск, перезавантаження або виключення системи та інші.

Обмеження безкоштовної версії програми TaskInfo 30 днів.

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

Окрім цього, TaskInfo також відображає детальну інформацію про операційну систему, папки Windows, про 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 налагоджувача процесів;
- звільнення фізичної оперативної пам'яті.

Microsoft Spy++ – це службова програма на основі Win32, що надає графічне подання системних процесів, потоків, вікон і вікон сповіщень.

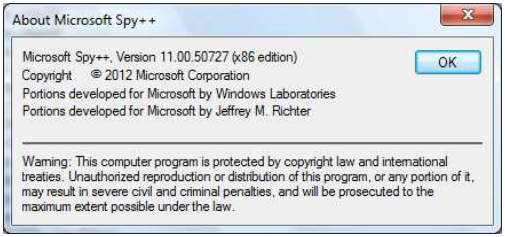


Рисунок 5.15 – Програма Microsoft Spy++

Існують дві версії Spy++. **Перша версія** з ім'ям Spy++ (*spyxx.exe*) призначена для відображення повідомлень, відправлених до вікна, що працює в 32-розрядному процесі. Наприклад, Visual Studio виконується у 32-розрядному процесі. Тому Spy++ можна використовувати для відображення повідомлень, відправлених на оглядач рішень**.** Оскільки конфігурація більшості версій Visual Studio за умовчанням призначена для 32-розрядного процесу, ця перша версія Spy++ є єдиною доступною в меню Інструменти в Visual Studio.

Друга 64-розрядна версія з ім'ям Spy++ (spyxx_amd64.exe) призначена для відображення повідомлень, відправлених до вікна, що працюють в 64-розрядному процесі. Наприклад, у 64-розрядній операційній системі Блокнот виконується в 64-розрядному процесі. Тому Spy++ (64-розрядна версія) можна використовувати для відображення повідомлень, відправлених до Блокнота. Звичайний шлях до Spy++ (64-розрядна версія):

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

Будь-яку з версій Spy++ можна запустити з командного рядка. Вікна подані у вигляді дерева (рис 5.16). Головним вікном є DeskTop, від нього йдуть розгалуження. У кожному вікні є ще вікна, і так до елементів управління. ОС Windows з цієї точки зору – дерево. Корінь – робочий стіл. Гілки – це запущені програми або процеси, а листя – це кінцеві вікна або елементи управління.

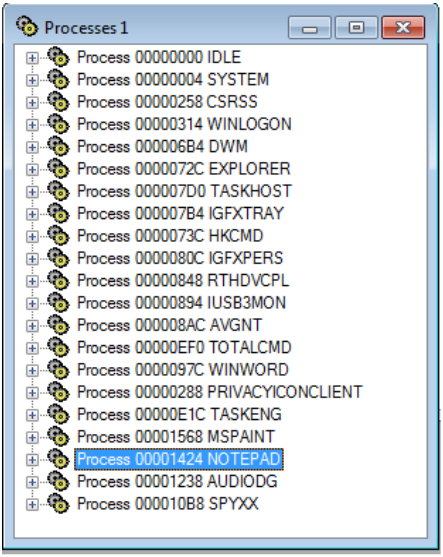


Рисунок 5.16 – Робоче вікно Spy++

Spy++ дозволяє виконувати такі завдання:

- відображати у вигляді дерева зв’язки між системними об’єктами, включаючи процеси, потоки і вікна;
- виконувати пошук вікон, що цікавлять, потоків, процесів і повідомлень;
- переглядати властивості вибраних вікон, потоків, процесів і повідомлень;
- вибирати вікна, потоки, процеси і повідомлення безпосередньо в поданні;
- використовувати інструмент пошуку для вибору вікна курсором миші;
- налаштовувати параметри повідомлень за допомогою параметрів вибору для комплексного ведення журналів повідомлень.

ЗАГАЛЬНЕ ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Провести, якщо необхідно, завантаження системного (призначеного для користувача) процесу, вказаного в **таблиці 5.2** індивідуального завдання, дослідженню властивостей якого і буде присвячена лабораторна робота.

До закінчення роботи індивідуальний процес НЕ ВИВАНТАЖУВАТИ з пам'яті

Таблиця 5.2 – Індивідуальні завдання для виконання

Вар.	Процес	Базовий пріоритет	Додаткове завдання
1	calc.exe	реального часу	Створюється додаток з управління рівнями пріоритетів потоків. Створити декілька потоків з різним пріоритетом. Користувач може вибрати рівень пріоритету довільного потоку. Вибір повинен здійснюватись у діалоговому вікні шляхом задання певної величини. Передбачити захист від неправильно введених даних
2	charmap.exe	високий	Написати програму, яка має довільне число вікон, в яких проводиться автоматичне рисування різних геометричних фігур. Проте рисування можливе тільки тоді, коли завантажений додаток notepad з порожнім документом. Якщо завершити роботу додатка notepad, то рисування повинно припиниться. При подальшому запуску notepad рисування поновлюється
3	cleanmgr.exe	вище середнього	Написати програму, яка при натисненні миші створює потоки: при натисненні правої клавіші – потік, що робить виведення зростаючого ряду, лівою – потік з убуваючим рядом. Потік вивантажується з пам'яті після закінчення рахунку. Кожному потоку поставити у відповідність своє вікно з випадковим кольором фону. Число потоків обмежується користувачем через контекстне меню і знаходиться в діапазоні від 1 до 100.
4	cliconfg.exe	середній	Створити багатопотокову програму, яка формує потоки трьох типів. Будь-який з потоків породжується відповідним пунктом меню і захоплює відповідно 1, 2, 3 одиниць умовного ресурсу (максимальні числа ресурсів за умовчанням – 10 і може змінюватися

Продовження табл. 5.2

Вар.	Процес	Базовий пріоритет	Додаткове завдання
			користувачем у вікні діалогу, який викликається через меню). Тут під ресурсом розуміють значення глобальної змінної. Кількість, вид потоків, а також їх стан виводиться на екран. Якщо число ресурсів не дозволяє працювати потоку, він знаходиться в стані очікування. Видалення потоків здійснюється через меню в порядку запуску (першим видалиться потік, запущений першим)
5	clipbrd.exe	нижче середнього	Запуск додатка повинен створити шість вікон, відповідних шести потокам: одне головне, останні – породжені, які у свою чергу залишаються в циклі введення аж до виходу. У кожне вікно організувати виведення унікального текстового повідомлення
6	cmstp.exe	низький	У програмі створити два потоки. Призначення одного з них – періодичне читання системного часу і заповнення глобальної структури (години, хвилини, секунди), другого – виведення даної структури на екран. Організувати роздільний доступ потоків до структури даних
7	ddeshare.exe	реального часу	Написати програму, яка містить два потоки. Кожному з потоків відповідає своє вікно і свої квіти. У меню НАЛАШТУВАННЯ програма повинна здійснити виведення зображення рисунка квітів (гвоздики, троянди, ромашки і так далі)
8	dxdiag.exe	високий	Написати програму, яка спочатку перераховує всі процеси, що виконуються в системі у вигляді списку з іменами і ідентифікаторами кожного процесу. Для поточного процесу повідомляється його ідентифікатор (разом з ідентифікатором батьківського процесу), клас пріоритету і кількість потоків, що виконуються зараз у контексті процесу
9	eudcedit.exe	вище середнього	Створити додаток з управління рівнями пріоритетів потоків. Створити декілька потоків з різним пріоритетом. Користувач може вибрати рівень пріоритету довільного потоку. При цьому вибір здійснювати за допомогою смуги прокручування. Після такого вибору виконати функцію, що інтенсивно використовує

Продовження табл. 5.2

Вар.	Процес	Базовий пріоритет	Додаткове завдання
			центральний процесор. Відобразити в робочій області отримані результати спільно з дескриптором потоку, рівнем пріоритету
10	eventvwr.exe	середній	Написати програму мініатюрної електронної таблиці, що складається з двох елементів: WRITER і ANSWER. У елемент WRITER можна записувати довільні числові значення (в т.ч. і дроби), а другий елемент ANSWER повинно бути реалізовано як статичний елемент управління, доступний тільки для читання. Поміщаючи число в поле WRITER, користувач примушує програму перерахувати значення в елементі ANSWER. Перерахунок полягає в тому, що лічильник, початкове значення якого дорівнює 0, поступово збільшується до максимуму, заданого в елементі WRITER. Для наочності виводити режим, в якому знаходиться програма: рахунок або очікування введення. Кожному режиму поставити у відповідність свій потік. Зміна числа в елементі WRITER у процесі рахунку повинна привести на початок відліку від нуля до введеного значення. Встановити крок рахунку рівний 0,1
11	magnify.exe	нижче середнього	Написати програму, що породжує 4 потоки, кожному з яких виділяється четверта частина вікна додатку. Перший потік виводить в свою область зведення часу створення потоку. Другий потік – у випадкові інтервали часу знищує перший або третій потік, а через 0,001 – 1 секунду відновлює видалений потік. Третій потік заповнює свою зону вікна іконою виконуваного модуля процесу. Четвертий потік фіксує в три зміни і виводить їх у своїй зоні вікна число запусків кожного з попередніх трьох потоків
12	mmc.exe	низький	Написати програму, яка при натисненні комбінації клавіш "CTRL + F2" створює потік, що робить виведення зростаючого ряду в позицію курсора вікна, комбінації клавіш "CTRL + F3" – потік з убуваючим рядом. Потік вивантажується з пам'яті після закінчення рахунку. Число потоків обмежується користувачем через контекстне меню. Кожному потоку поставити у відповідність своє вікно

Продовження табл. 5.2

Вар.	Процес	Базовий пріоритет	Додаткове завдання
13	mobsync.exe	реального часу	Передбачити введення користувачем як параметр роботи додатка довгого значення змінною. Другий пункт меню повинен мати такі підпункти: «Збільшити», «Зменшити», «Обмін». Після чого запустити декілька екземплярів даного додатка. Довільний вибір операцій збільшення (зменшення) приводить до зміни вибраної змінної, і відображення її значення в робочій області кожного з екземплярів додатка. Крім того, там же відображається значення дескрипторів створених вікон запущених екземплярів додатків
14	mshearts.exe	високий	Створюється додаток з управління рівнями пріоритетів потоків. Створити декілька потоків з різним пріоритетом. Користувач може вибрати рівень пріоритету довільного потоку. Вибір повинен здійснюватися в діалоговому вікні шляхом задання певної величини. Передбачити захист від неправильно введених даних
15	msiexec.exe	вище середнього	Створити додатковий пункт меню НАЛАШТУВАННЯ, в якому користувач повинен задати максимально допустимий час (у секундах) знаходження тестового потоку і максимальне число таких тестових потоків. Кожному потоку повинне відповідати своє вікно з випадковим кольором фону. При виборі третього пункту головного меню повинні створюватися нові потоки (і відповідні їм вікна), які через 5 секунд мають бути завершені
16	mstsc.exe	середній	При виборі другого пункту меню запускається другий процес, причому обмежити можливість подальшого запуску процесів у цього пункту. Задання другого процесу стежити за натисненнями функціональних клавіш. У робочій області відображається найменування натиснутої клавіші, а в другому вікні – факт натиснення у вигляді символу «*». Таким чином, один процес стежить за введенням даних в іншому процесі. Передбачити можливість закриття одного процесу з меню іншого процесу
17	narrator.exe	нижче середнього	Написати програму, яка породжує потік з натиснення однієї з функціональних клавіш клавіатури. Кожен створений у такий спосіб

Продовження табл. 5.2

Вар.	Процес	Базовий пріоритет	Додаткове завдання
			потік рисує круг у вікні додатку. Досягши межі вікна, круг змінює напрям свого проходження на протилежне. У разі руху декількох кругів пріоритет руху має бути у потоку, який створений раніше
18	netsetup.exe	низький	Запуск додатка повинен створити два вікна, відповідні двом потокам: головне і породжене, який у свою чергу залишається в циклі введення аж до виходу з програми. Коли користувач вибирає другий пункт головного меню, програма повинна перемикатися на приєднаний стан введення породженого потоку, а також встановити активне вікно як вікно іншого потоку. Ця операція має бути успішною, якщо механізми введення обох потоків сполучені, і невдалою, якщо вони роз'єднані. Результат супроводжувати відповідними повідомленнями в робочих областях вікон
19	notepad.exe	реального часу	Написати програму, що дозволяє експериментувати з класами пріоритетів процесів і досліджувати їх 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	nslookup.exe	високий	Написати програму, яка породжує потік при натисненні однієї з функціональних клавіш клавіатури. За кожною з функціональних клавіш закріплюється свій графічний об'єкт – (круг, еліпс, прямокутник і так далі). Кожен створений у такий спосіб потік рисує замальований графічний об'єкт у вікні додатка. Пріоритет при рисуванні буде у того потоку, який раніше був задіяний. Нові геометричні фігури з'являтимуться попереду вже нарисованих.
21	osk.exe	вище середнього	Написати програму, що дозволяє експериментувати з класами пріоритетів процесів і досліджувати їх вплив на загальну продуктивність системи. При натисненні на праву кнопку миші створюється новий процес. У робочу зону вікна процесу виводиться PID

Продовження табл. 5.2

Вар.	Процес	Базовий пріоритет	Додаткове завдання
			процесу і його дескриптор. Спочатку перший процес у своє вікно виводить значення, що збільшується на одиницю. Одним з пунктів меню проводити зміну класу пріоритету поточного процесу (табл. 5.1). У відповідне вікно виводити повідомлення про новий пріоритет. Натиснення на будь-який з процесів лівої клавіші приводить до його завершення
22	packager.exe	середній	Написати програму, що дозволяє експериментувати з відносними пріоритетами потоків і досліджувати їх вплив на загальну продуктивність системи. Спочатку первинний потік працює дуже активно, і ступінь використання процесора підскакує до 100 %. Все, чим він займається, – постійно збільшує початкове значення довільної змінної на 1 і виводить поточне значення у вікно. Передбачити зміну рівня пріоритету потоку
23	perfmon.exe	нижче середнього	Написати програму, яка запускає новий потік при натисненні лівої клавіші миші. Потік починає виводити зростаючу числову послідовність в робочу зону вікна. При натисненні лівої клавіші миші програма видаляє потік
24	shrpubw.exe	низький	Написати додаток, в якому створюється мінімум чотири потоки: два основних, два – додаткових. Основні потоки проводять в своєму вікні виведення випадкових букв, додаткові потоки – геометричні фігури (мінімум 6) випадкового кольору і розташування
25	sigverif.exe	реального часу	Коли користувач вибирає в додатку другий пункт меню, створюється потік, який виводить довільне текстове повідомлення в робочу зону вікна, яке поступово заповнює її дати, поки не буде натиснута комбінація клавіш "CTRL + F9". Натиснення цієї комбінації повинно припинити виведення. Повторне натиснення повинно відновлювати виведення
26	sysedit.exe	високий	Написати програму, яка запускає новий потік при натисненні правої клавіші миші. Потік починає виводити довільні числа із заданого користувачем діапазону в робочу зону вікна. При натисненні

Закінчення таблиці 5.2

Вар.	Процес	Базовий пріоритет	Додаткове завдання
			лівої клавіші миші програма видаляє створений потік. Передбачити обмеження повторного натиснення правої кнопки миші
27	taskmgr.exe	вище середнього	Написати додаток, в якому один потік створюватиме інший потік. Цей потік формує відомості про: - тип процесу (64- або 32-бітовий); - базовий пріоритет виконання процесу; - загальна кількість потоків (threads) для кожного процесу; - поточне значення номера потоку процесу. Перший потік повинен знаходитися в стані очікування на період, що задається користувачем (від 0,001 до 50 сек.), після чого закриває другий потік
28	verifier.exe	середній	Створити додатковий пункт меню НАЛАШТУВАННЯ, в якому користувач повинен задати максимально допустимий час (у секундах) знаходження тестового потоку і максимальне число таких тестових потоків. Кожному потоку повинне відповідати своє вікно з випадковим кольором фону. При виборі третього пункту головного меню повинні створюватися нові потоки (і відповідні йому вікна), які через випадковий час, але не більше максимально допустимого, мають бути завершені
29	write.exe	нижче середнього	Написати програму, що дозволяє експериментувати з класами пріоритетів процесів і досліджувати їх вплив на загальну продуктивність системи. Первинний процес повинен активно використовувати процесор, завантаживши його виконанням безлічі обчислень. Передбачити зміну класу пріоритету процесу (табл. 5.1). Користувач, використовуючи деяке поле ПРИПИНЕННЯ ПРОЦЕСУ, може припинити первинний потік на задане число мілісекунд у діапазоні від 0 до 9999
30	wscript.exe	низький	Написати додаток, який реалізує мультизадачний додаток MultiMDI, суть якого полягає в тому, що в головному вікні додатка створити дочірні вікна, в яких виконуються циклічне відображення еліпсів випадкової форми і кольору, і реалізувати можливість установлювати відносний пріоритет окремих завдань, припиняти їх, відновлювати припинені завдання, видаляти завдання

Етап 1. Провести дослідження індивідуального процесу за допомогою програми *Диспетчера завдань*. При цьому встановити для нього базовий пріоритет, указаний в таблиці 5.2 індивідуального завдання. Результати зафіксувати у вигляді таблиці 5.3. Провести підтвердження результатів скриншотами.

Таблиця 5.3 – Результати дослідження процесу в *Диспетчері завдань*

Ім'я образу	PID	Базовий пріоритет	Максимальний обсяг фізичної пам'яті процесу	Обсяг віртуальної пам'яті процесу

Етап 2. Провести дослідження індивідуального процесу за допомогою програми *Aida*. Результати зафіксувати у вигляді таблиці 5.4. Провести підтвердження результатів відповідними скриншотами.

Таблиця 5.4 – Результати дослідження процесу в *Aida*

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

Етап 3. Провести дослідження індивідуального процесу за допомогою програми *Process Explorer*. Результати зафіксувати у вигляді таблиці 5.5. Якщо у процеса декілька потоків, то перерахувати усі їх TID. Провести підтвердження результатів відповідними скріншотами.

Таблиця 5.5 – Результати дослідження процесу в *Process Explorer*

Процес	PID	Базовий пріоритет	Максимальний обсяг фізичної пам'яті процесу	Обсяг віртуальної пам'яті процесу	Дескриптори		TID	Динамічний пріоритет
					системних	users		

Етап 4. Побудувати за допомогою програми *Process Monitor* дерево процесів у Вашій операційній системі, причому особливу увагу приділити дослідженню дерева для індивідуального процесу. Наприклад, для процесу Wintest.exe дерево має вигляд, зображений на рис. 5.17.

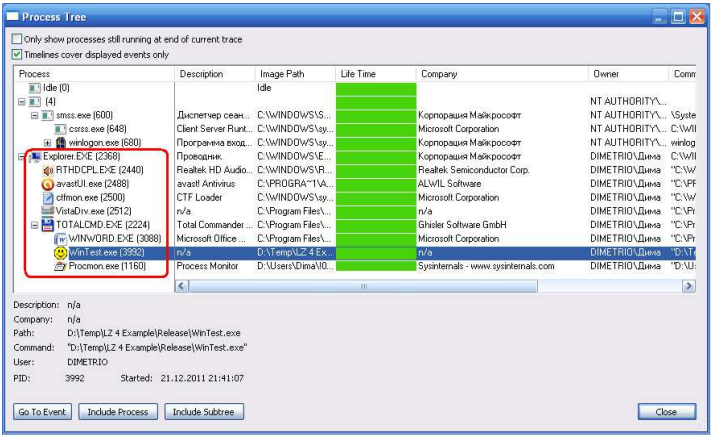


Рисунок 5.17 – дерево процесів для процесу Wintest.exe

Етап 5. Провести дослідження індивідуального процесу за допомогою програми *System Info for Windows*. Результати зафіксувати у вигляді таблиці 5.6. Провести підтвердження результатів відповідними скриншотами.

Таблиця 5.6 – Результати дослідження процесу в *System Info for Windows*

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

Етап 6. Провести дослідження індивідуального процесу за допомогою програми *Task Info*. Результати зафіксувати у вигляді таблиці 5.7. Якщо у процесу декілька потоків, то перерахувати усі їх TID. Провести підтвердження результатів відповідними скриншотами.

Таблиця 5.7 – Результати дослідження процесу в *Task Info*

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

Етап 7. Провести дослідження індивідуального процесу за допомогою програми *Microsoft Spy++*. Результати зафіксувати у вигляді таблиці 5.8. Провести підтвердження результатів відповідними скриншотами.

Таблиця 5.8 – Результати дослідження процесу в *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

Створити програмний продукт у середовищі Visual Studio, в якому передбачити наявність мінімум двох пунктів головного меню. **Перший** пункт меню призначений для виведення статистичних даних про поточний процес (потік) у вигляді єдиного діалогового вікна параметрів:

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, що надано у таблиці 5.2.

Контрольні запитання

1. Поясніть фізичний сенс поняття «Базовий пріоритет процесу».
2. У чому полягають принципові відмінності програм моніторингу процесів і потоків?
3. Які відмінні характеристики процесів дозволяють відображати програми моніторингу?
4. Назвіть виробників використаних програмних продуктів.
5. Наведіть приклади використання програм при боротьбі з вірусами.
6. Обгрунтуйте переваги і недоліки програм діагностики процесів.
7. Назвіть основні характеристики процесів і потоків у Windows.

Лабораторна робота 6

ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ ВІРТУАЛЬНОЇ ПАМ'ЯТІ

Мета роботи: ознайомитись з побудовою і організацією механізму віртуальної пам'яті, дослідити стани сторінок віртуальної пам'яті, типи доступу до сторінок віртуальної пам'яті, крім того, вивчити способи отримання з кучі пам'яті невеликих розмірів для використання додатка.

Указівки з підготовки до виконання лабораторної роботи

Необхідно вивчити основні елементи організації пам'яті. Система управління пам'яттю є найскладнішою і найдосконалішою. Від того, наскільки добре будуть відпрацьовані програмні інтерфейси цієї системи, багато в чому залежить ефективність і працездатність створюваних застосувань. Додаткову інформацію при підготовці до роботи можна отримати з літератури:

1. Побегайло А. П. Системное программирование в Windows / А. П. Побегайло – СПб.: БХВ-Петербург, 2006. – 1056 с.

2. Саймон Р. Windows 2003 API. Энциклопедия программиста: пер. с англ. /Р.Саймон. – Киев : ООО "ДиасофтЮП", 2004. – С.340 – 373.

3. Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows: пер. с англ. – СПб.: Питер, 2006. (Глава 14).

Теоретичні відомості

У середовищі Windows кожен процес має власний 32-розрядний віртуальний адресний простір обсягом до 4 Гбайт. Користувачеві доступні лише 2 Гбайт в ділянці нижньої пам'яті (від 0x00000000 до 0x7FFFFFFF), а 2 Гбайт у ділянці верхньої пам'яті (від 0x80000000 до 0xFFFFFFFF) зарезервовано для ядра ОС.

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 пере-

міщати пам’ять і управляти нею з максимально можливою ефективністю з урахуванням існуючих 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’яті та кучу.

Глобальна пам’ять розподіляється з глобальної кучі процесу. В більшості випадків цим дескриптором є вказівник на розподілену пам’ять. Пам’ять розподіляється у вигляді приватних, виділених сторінок з 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 пам’яті. Коли система переміщає сторінку пам’яті, віртуальна сторінка процесу просто відображається в інше місце. Переміщувана пам’ять все 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. У цілях збільшення розмірів пам’яті в системі для розподілу додаткової пам’яті застосовується жорсткий диск. Сумарний обсяг пам’яті, доступної для всіх процесів, є сумою обсягу фізичної пам’яті і величини вільного простору на диску, доступного для файлу підкачки (файлу на диску, вживаного для збільшення обсягу пам’яті). Віртуальний адресний простір організований у вигляді сторінок або одиниць обсягу 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 обсягів пам’яті.

Куча. Функції *кучі* дозволяють процесу створювати приватну кучу – блок з

однієї або декількох сторінок в адресному просторі процесу. Пам'ять у приватній кучі доступна тільки для процесу, що створив кучу.

Для дослідження глобальної, віртуальної пам'яті і кучі існує дві основні групи програмних засобів: вбудовані в ОС і сторонніх виробників.

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

1. *Диспетчер пристроїв Windows* (Device Manager Windows).

2. *Монітор ресурсів* (Resource monitor) – для Windows 7.

До програм сторонніх виробників, розгляду яких присвячена лабораторна робота, відносяться:

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 (by Golubnychiy).*

Диспетчер пристроїв. Використовується для оновлення драйверів обладнання, зміни налаштувань обладнання, а також для усунення несправностей (рис. 6.1). У Windows пристрої класифікуються за типами. Типи пристроїв включають: плати відеоадаптерів, клавіатури, пристрої читання компакт-дисків, порти і принтери. У вікнах диспетчера пристроїв і майстра установлення обладнання відображається список типів пристроїв, установлених на даному комп'ютері.

Типи пристроїв у свою чергу підрозділяються на категорії відповідно до конкретних пристроїв. Наприклад, тип «Модеми» включає сотні різних модемів, які можна встановити і використовувати з Windows.

Пристрої класифікуються також за способом їх підключення до комп'ютера. Більшість з них постійно підключена до комп'ютера і встановлюється тільки один

раз. Такі пристрої доступні при кожному запуску комп'ютера, якщо тільки вони не відключені або не видалені. Прикладами можуть служити звукові плати, плати відеоадаптерів, модеми і жорсткі диски.

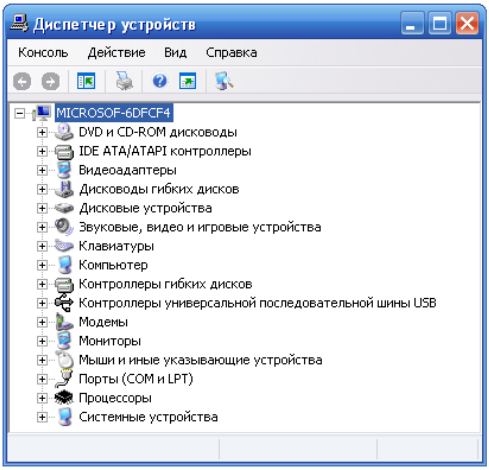


Рисунок 6.1 – Диспетчер пристроїв

Інші пристрої розраховані на підключення до комп'ютера і відключення від нього за необхідності. Вони можуть підключатися до відповідного порту або гнізда розширень, і Windows розпізнає, налаштовує їх без перезавантаження комп'ютера. При відключенні таких пристроїв необхідно лише повідомляти Windows про їх витягання, видалення або відключення. Вимикати або перезапускати комп'ютер не потрібно. Пристроями цього типу є:

1. Плати PC, що підключаються до переносних комп'ютерів.

2. Пристрої, що підключаються до шини USB або шині IEEE 1394.

3. Стикувальні вузли, що підтримують гаряче стикування і відстикування переносних комп'ютерів.

4. Пристрої, що підключаються 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 сучаснішої версії призначеним для користувача інтерфейсом і обмежений у функціональності. У Windows XP для відстежування ресурсів використовується Диспетчер завдань (Task Manager).

Запустити Монітор ресурсів можна декількома способами. Якщо вже відкритий Диспетчер завдань, перейдіть на вкладку «Швидкодія» (Performance) і натисніть кнопку «Монітор ресурсів» (рис. 6.2). Його також можна викликати з меню «Пуск → Всі програми → Стандартні → Службові» або просто ввести **Resmon.exe** у рядку виконання команд і натиснути Enter.

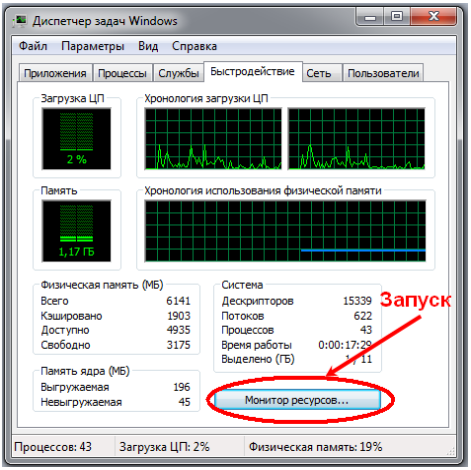


Рисунок 6.2 – Запуск Монітору ресурсів

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

З погляду лабораторної роботи нам цікава буде третя вкладка *Пам’ять* (*Memory*) (рис. 6.4). На ній наводяться детальні відомості про використання пам’яті. Додатково в розділі «Фізична *пам’ять*» (Physical *Memory*) є унікальна гістограма, що відображає розподіл пам’яті.

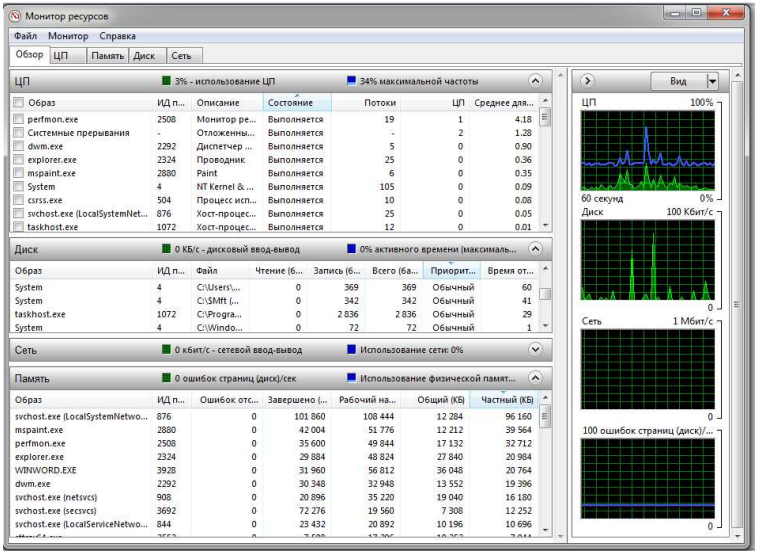


Рисунок 6.3 – Вікно Монітора ресурсів

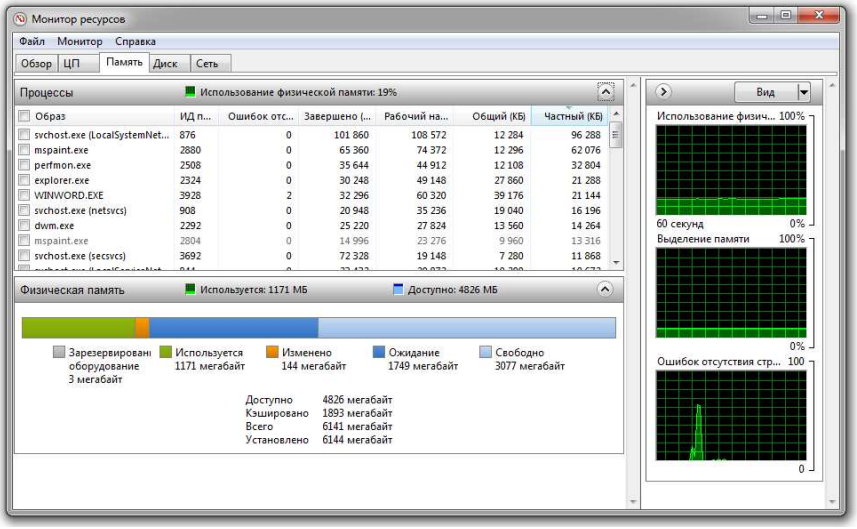


Рисунок 6.4 – Вкладка «Пам’ять» у вікні Монітора ресурсів

Справа на кожній вкладці розташовані графіки. Вони безперервно оновлю-

ються і відображають стан за останню хвилину. Щоб вивчити певну активність детальніше, перш ніж ця ділянка графіка сховається, можна вибрати команду «Зупинити моніторинг» (Stop Monitoring) в меню «Монітор» (Monitor). Поновлюється моніторинг командою «Запустити моніторинг» (Start Monitoring).

При виборі певного процесу всі інші фільтруються, завдяки чому стає набагато простіше зрозуміти, як саме даний процес впливає на розподіл ресурсів.

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

Таблиця «Процеси». На вкладці «Пам'ять» є таблиця «Процеси» (Processes), в якій перераховані всі запущені процеси, а відомості про використовувану пам'ять розбиті на декілька категорій (див. рис. 6.4):

− **Графа «Образ» (Image).** Вказується ім'я виконуваного файлу процесу.

− **Графа «ІД процесу» (PID).** Вказується номери процесу – унікальне поєднання цифр, що дозволяє ідентифікувати запущений процес.

− **Графа «Завершено» (Commit).** Вказуються обсяг віртуальної пам'яті у кілобайтах, зарезервований системою для даного процесу. Сюди входить і фізична пам'ять, що використовується, збережені у файлі підкачки сторінки.

− **Графа «Робочий набір» (Working Set).** Вказується обсяг фізичної пам'яті в кілобайтах, що використовується процесом у даний момент часу. Робочий набір складається з загальної і приватної пам'яті.

− **Графа «Загальний» (Shareable).** Вказаний обсяг фізичної пам'яті в кілобайтах, яку даний процес використовує спільно з іншими. Використання одного сегмента пам'яті або сторінки підкачки для споріднених процесів дозволяє заощадити місце в пам'яті. При цьому фізично зберігається тільки одна копія сторінки, яка потім зіставляється з віртуальним адресним простором інших процесів, які до неї звертаються. Наприклад, усі процеси, які ініційовані системними бібліотеками DLL – Ntdll, Kernel32, Gdi32 і User32 – використовують загальну пам'ять.

– **Графа «Приватний» (Private).** Вказується об’єм фізичної пам’яті в кілобайтах, яка використовується виключно даним процесом. Саме це значення дозволяє визначити, скільки пам’яті потрібно тому або іншому додатку для роботи.

– **Графа «Помилок відсутності сторінки в памяті/с.» (Hard Faults/sec).** Вказана середня за останню хвилину кількість помилок відсутності сторінки в пам’яті в секунду. Якщо процес намагається використати більше фізичної пам’яті, чим доступно в даний момент часу, система записує частину даних з пам’яті на диск – у файл підкачки. Подальше звернення до даних, збережених на диск, і називається помилкою відсутності сторінки в пам’яті.

Таблиця «Фізична пам’ять». У таблиці "Процеси" наводяться 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. Кожна секція гістограми позначена власним кольором і становить 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 розташована секція "Зарезервоване обладнання" (Hardware Reserved), позначена 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 обсяг пам’яті, виділений пристрою, становить від 1 до 70 Мбайт, проте цей показник залежить від конкретної конфігурації системи і в деяких випадках може досягати декількох сотень мегабайт. До компонентів, що впливають на обсяг зарезервованої пам’яті, належать:

– BIOS;

– компоненти материнської плати — наприклад, вдосконалений програмований контролер переривань введення/виведення (APIC);

– звукові карти та інші пристрої, що здійснюють введення/виведення з відображенням на пам’ять;

– шина PCI Express (PCI-E);

– відеокарти;

– різні набори мікросхем;

– флеш-накопичувачі.

Секція «Використовується». Секція «*Використовується*» (In Use), позначена зеленим кольором, становить кількість пам’яті, що використовується системою, драйверами та запущеними процесами. Кількість використовуваної пам’яті розраховується як значення «*Всього*» (Total), за винятком суми показників «*Змінене*» (Modified), «*Очікування*» (Standby) і «*Вільно*» (Free). У свою чергу, значення «*Всього*» – це показник «*Встановлено*» (Installed RAM), за винятком показника «*Зарезервоване обладнання*».

Секція «Змінено». Помаранчевим кольором виділена секція «*Змінено*» (Modified), в якій подана змінена, але не задіяна пам’ять. Фактично вона не використовується, але може бути у будь-який момент задіяна, якщо знову знадобиться. Якщо пам’ять не використовується достатньо давно, дані переносяться у файл підкачки, а пам’ять переходить в категорію «*Очікування*».

Секція «Очікування». Позначена синім кольором, становить сторінки пам’яті, видалені з робочих наборів, але як і раніше з ними зв’язані. Іншими словами, категорія «*Очікування*» – це фактично кеш. Сторінкам пам’яті в цій категорії присвоюється пріоритет від 0 до 7 (максимум). Сторінки, пов’язані з високопріоритетними процесами, отримують максимальний пріоритет. Проте всі сторінки в категорії «*Очікування*» доступні для запису даних від інших процесів.

Секція «Вільно». Позначена блакитним кольором, де подані сторінки пам’яті, ще не виділені жодному процесу або які звільнилися після завершення процесу. У цій секції відображається як ще не задіяна, так і вже звільнена пам’ять, але насправді, ще не задіяна пам’ять належить до іншої категорії – «*Нульові сторінки*» (Zero Page), яка так називається тому, що ці сторінки заповнені нульовим

значенням і готові для використання.

FreeRAM (by BySoft) – це простий менеджер пам'яті з маленьким розміром дистрибутиву (рис. 6.5). Інтерфейс досить простий і має декілька індикаторів (завантаження процесора, кількість вільної пам'яті і т.д.), включаючи графік використання пам'яті. Менеджер має підтримку обкладинок, серед яких є обкладинки невеликого розміру, це дозволяє не ховати менеджер і відстежувати інформацію про стан пам'яті. У налаштуваннях можна налаштувати автоматичне звільнення пам'яті за декількома умовами. Так само можна налаштувати звільнення пам'яті гарячим клавішам і встановити звукове сповіщення після звільнення пам'яті.

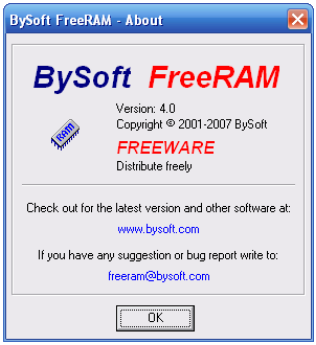


Рисунок 6.5 – Програма FreeRAM (by BySoft)

Характеристики:

– простота у використанні;

– ручна і автоматична оптимізація пам'яті;

– скидає на диск невживані бібліотеки;

– підвищує ефективність вашого комп'ютера;

– звільняє певну кількість пам'яті натисненням кнопки;

– звільняє заздалегідь певну кількість пам'яті (5 %, 10 % ... 80 %);

– відображає графік вільної пам'яті і використовує центральний процесорний пристрій в режимі реального часу;

– автоматично або вручну налаштовує системний файловий кеш;

– підтримує змінні оболонки;

– використовує системний tray.

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

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

Якщо перша група параметрів *Automatic* включена, то буде проводитись автоматичне звільнення пам'яті досягнувши критичної відмітки за значенням, що визначається параметром «Обсяг вільної пам'яті (МБ)» (Amount *of memory free*(*MB*)) (рис. 6.6). При цьому кількість спроб звільнення пам'яті не перевищуватиме значення, що визначається параметром *Number of retries if desired amount is not freed.*

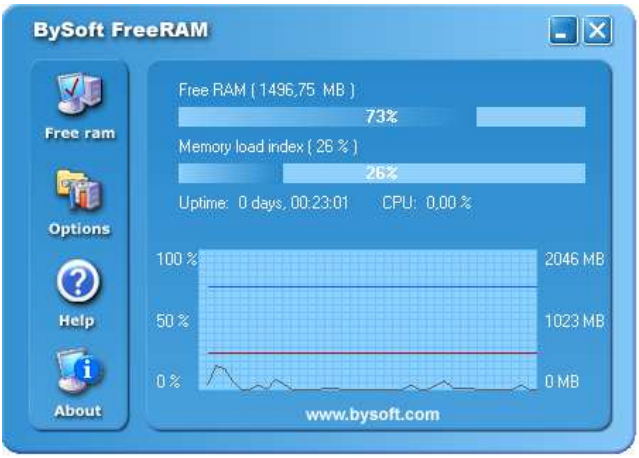


Рисунок 6.6 – Робоче вікно FreeRAM

Друга група параметрів (Setting) визначає порядок відображення програми:

поверх інших вікон (*Stay on top*), запускати після запуску Windows (*Start with Windows*), запускати мінімізованою в трей (*Start minimized*) і так далі.

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

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

Mz RAM Booster v.4.1 (by Michael Zacharias). Невеликий безкоштовний додаток для очищення оперативної пам'яті комп'ютера, таким чином, збільшує швидкість роботи системи.

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. Даний засіб оптимізує роботу оперативної пам'яті за рахунок завершення процесів, що знаходяться в стані простою, або зупинки фонових завдань. Користувач зможе без ризику для стабільної роботи системи зробити роботу оптимальною, підвищивши продуктивність.

У головному вікні програми можна отримати основну інформацію про завантаженість фізичної пам'яті, а графік допоможе наочно оцінювати зміну завантаженості. Надано основні значення про використання віртуальної пам'яті, тобто розмір і зайнятість файлу підкачки. Для загальної інформації автор додав індикатор загальної завантаженості центрального процесора і можливість вплинути на інтервал оновлення даних. Для зручності в правій частині вікна реалізовані швидкі кнопки для запуску основних завдань (рис. 6.7).

Вкладка System SpeedUp призначена для зміни прискорення роботи системи. Параметрів для коректування небагато, але всі вони дійсно належать до оптимізації комп'ютера, а саме:

− блок функції Windows для створення штампу за останнім часом доступу до NTFS файлів (Disable NTFS Last Access Update Stamp);

– відключити підтримки створення імен формату 8.3 (Disable 8.3 Filename Creation);

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

– вивантажувати автоматично бібліотеки DLL, які не використовувались (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).

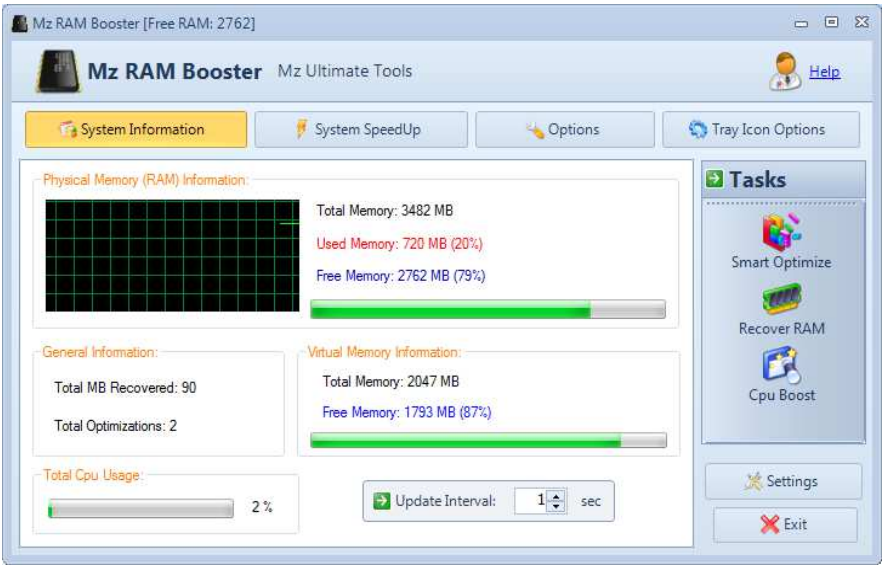


Рисунок 6.7 – Робоче вікно Mz Ram Booster

FreeRAM XP Pro 1.52 (by M.Xiang) – менеджер оперативної пам’яті комп’ютера (рис. 6.8). Ця утиліта дозволяє провести дефрагментацію і звільнити оперативну пам’ять під час роботи комп’ютера. Тим самим збільшується швидкість і стабільність системи в цілому. 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 присутній ряд інформаційних інструментів.

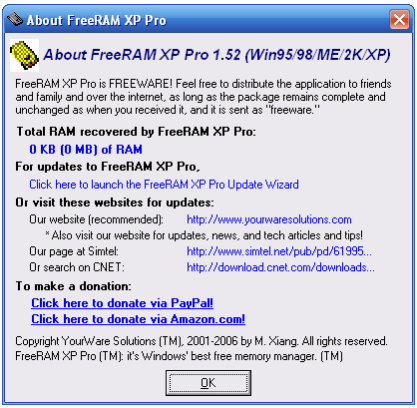


Рисунок 6.8 – Про програму FreeRAM XP Pro 1.52

Інструмент Report Process Memory Usage дозволяє відобразити обсяг зайнятої пам'яті кожного з процесів (додатків), запущених у системі (рис. 6.9). FreeRAM XP Pro не вимогливий до програмних і апаратних ресурсів комп'ютера. Окрім двох стандартних індикаторів завантаження пам'яті, передбачені індикатори завантаження віртуальної пам'яті і процесора. Загальні відомості про основні (base) та розширені (advanced) параметри оперативної пам'яті, кеш, файлу підкачки і віртуальної пам'яті містяться у звіті File → Generate Memory Report.

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

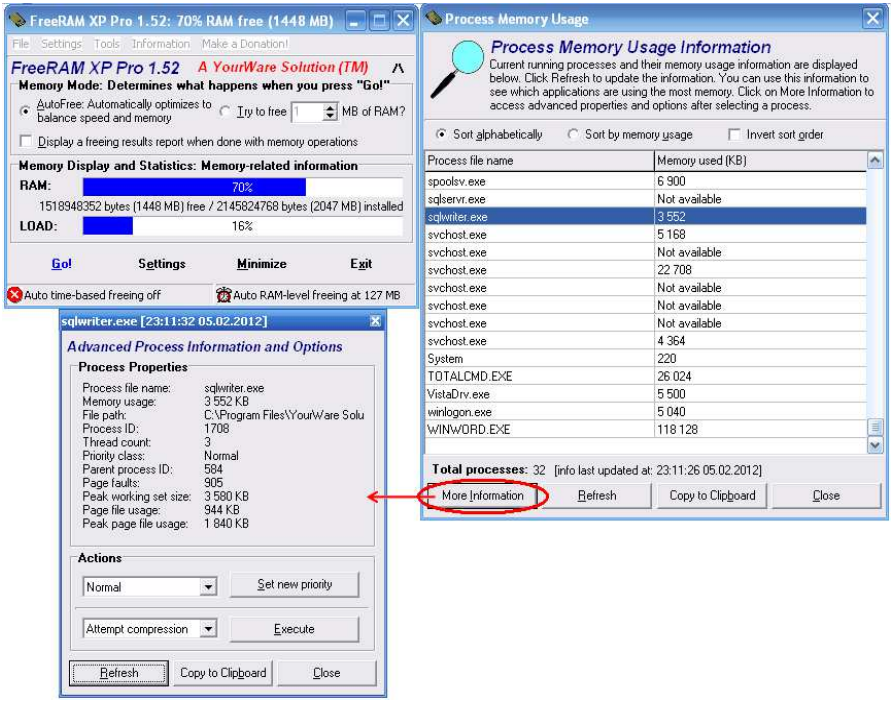


Рисунок 6.9 – Інструмент Report Process Memory Usage

Вибір процесу, що цікавить, дозволить отримати розширений набір таких параметрів: ім'я файлу процесу; обсяг пам'яті, що використовується; шлях до файлу; ідентифікатор процесу; кількість потоків; клас пріоритету процесу; ідентифікатор батьківського процесу; кількість сторінок помилок; максимальний робочий обсяг процесу; ємність використаної частини процесом файлу підкачки; максимальний обсяг використаного файлу підкачки.

Утиліта підтримується всіма версіями Windows: від Windows 98 до Windows 8.

CleanMem v.2.4.3 (by PcWinTech.com). Утиліта **CleanMem** є непоганим оптимізатором оперативної пам'яті. Актуальність застосування програми полягає в необхідності використання на комп'ютерах користувачів з невеликим обсягом оперативної пам'яті (рис. 6.10).

Після запуску цієї утиліти можна значно вивільнити оперативну пам’ять, тим самим зменшити очікування ресурсомістких процесів, запущених в операційній системі, буде звільнено пам’ять для інших процесів, що мають потребу.

Рисунок 6.10 – CleanMem v.2.4.3 (by PcWinTech.com)

CleanMem достатньо проста у використанні, швидко інсталюється та запускається. Управління можливе як і за допомогою контекстного меню в системному Tray, так і за допомогою пункту *Menu* графічної оболонки Mini Monitor Free.

Варто відзначити, що CleanMem дуже мало важить, майже не завантажує систему, не витрачає оперативної пам’яті. Запускати програму можна на будь-якому комп’ютері, переносити з собою її можна на будь-якому накопичувачі. CleanMem запросто може провести оптимізацію пам’яті, робиться це за допомогою стандартного інтерфейсу Windows. Програма має досить гнучкі налаштування, наприклад використовуючи спеціальні сценарії CleanMem можна легко змусити програму ігнорувати один процес і включати в процес оптимізації всі останні, тобто можна виділити ті, які чіпати не потрібно. Є можливість запуску утиліти за потрібним розкладом, програма веде звіти, які можна потім переглядати.

RAM Saver Professional v.11.11. Інструмент для професійного моніторингу, очищення і оптимізації оперативної пам’яті. Служить для підвищення продуктив-

ності операційної системи, звільняючи оперативну пам’ять від драйверів і процесів MS Windows для додатків, що вимагають максимального завантаження процесора і оперативної пам’яті.

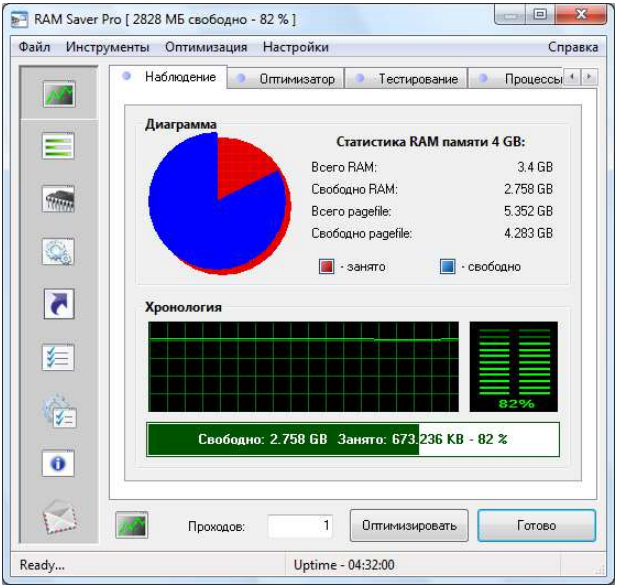


Рисунок 6.11 – RAM Saver Professional

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 за одним натисненням;
- відображення часу з моменту включення комп'ютера;
- придушення і швидкий запуск заставки;
- перевірка наявності компакт-диска в CD-ROM приводі;
- можливість приховати всі ікони робочого столу;
- примусове виключення і перезавантаження комп'ютера.

VMMap v.3.11 (by Mark Russinovich and Bryce). Утиліта для діагностики несправності на основі споживання системної пам'яті: дозволяє отримати візуальну карту розподілу фізичної та віртуальної пам'яті. 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'яті, керовану кучу, стік і системну пам'ять.
4. Дозволяє уникнути великих поломок.
5. VMMap абсолютно безкоштовна утиліта.

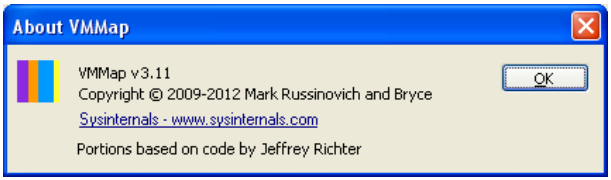


Рисунок 6.11 – Про прогаму VMMap

Використання VMMap зовсім нескладне: після установлення та запуску можна вибрати в меню один із запущених процесів для складання карти пам'яті.

На карті показана різниця в обсязі переданої пам'яті і робочої множини, 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, куча (heap), керована куча, стек і системна пам’ять (рис. 6.12).

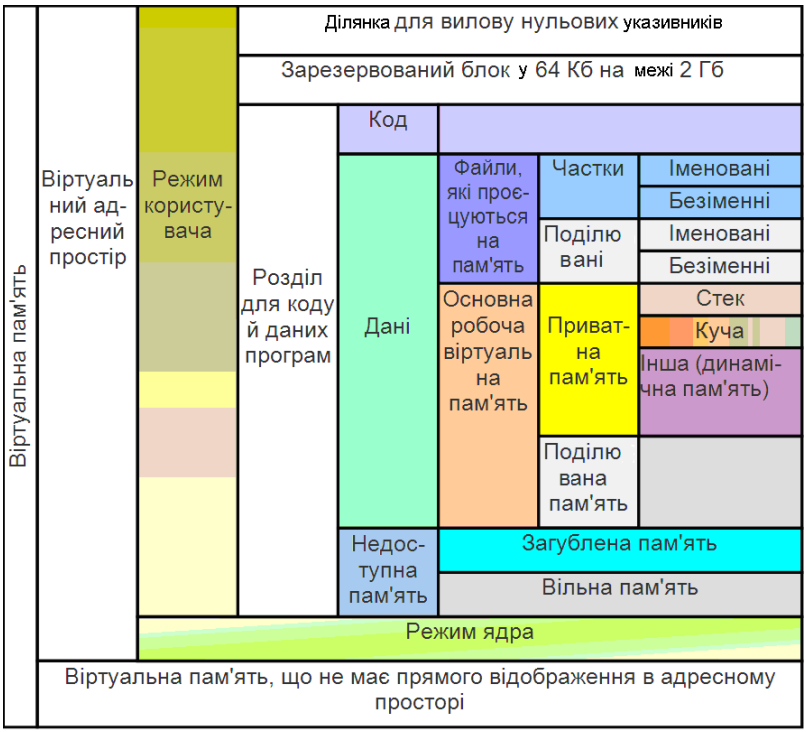


Рисунок 6.12 – Загальна схема класифікації різних блоків в адресному просторі

Зона секції коду (Image). Помічена 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 і стає його частиною. Будь-яке звернення до пам’яті за цими адресами приведе до фактичного читання даних з диска. З точки зору програми ця частина виглядає як звичайна «пам’ять», нічим не відрізняється від будь-якої іншої пам’яті. Але насправді ця пам’ять читається з файлу на диску.

Атрибути захисту (доступу) виставляються 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 комбінуватись в будь-якій послідовності (відсутність атрибутів можна вважати як атрибут «No access» – PAGE_NOACCESS), а також два спеціальні модифікатори – copy-on-write і guard.

Все, 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). Вони помічені синім кольором. Проектування файлів – це зручний спосіб обробки даних. Передбачається, що дані знаходяться в пам'яті, хоча насправді вони розташовані на диску. Програмістові не потрібно писати явний код завантаження файлу в пам'ять, заощаджується час на переміщенні блоків пам'яті. Все 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 із сторінкового файлу, а береться з файлу, що вже знаходиться на диску. Як тільки файл спроектований в пам'ять, до нього можна звертатися так, ніби він цілком в неї завантажений.

Приватна (private) пам'ять та пам'ять, яка розділяється (shareable). Вони помічені жовтим і блакитним кольорами відповідно. Більшу частину займають звичайні дані, які індивідуальні для програми користувача. Це так звані private (приватні) дані. Ці дані використовуються тільки нашою 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 розділяються) дані. Це спеціальним чином оформлені секції DLL-файлів, які спільно

використовуються декількома програмами.

Втрачена (unusable) і вільна (free) пам'ять. Вони помічені сірим і білим кольорами відповідно. Перераховані вище основні регіони – все, що можна використовувати. Вся інша пам'ять є недоступною. Будь-яке звернення до неї призводить до виключення порушення доступу (Access Violation) – це окремий випадок неправильного доступу пам'яті. Але навіть недоступна пам'ять ділиться на дві категорії. Найочевидніша – вільна пам'ять. Її більшість.

Окрім вільної пам'яті, в недоступній пам'яті значиться пам'ять, яка не може бути використана через фрагментацію. Пам'ять виділяється блоками по 64 Кб, але гранулярність блоку пам'яті – розмір сторінки, тобто 4 Кб. Це означає, що якщо треба виділити 2 Кб, то ОС виділить 4 Кб, а 60 Кб виявляться навіки недоступними – поки не звільняться ці 2 Кб (насправді – 4 Кб) пам'яті. Отже, весь блок у 64 Кб не стане знов цілком доступним для використання.

Системна куча (heap). Вона помічена червоним кольором. Куча – це назва структури даних, за допомогою якої реалізований динамічний розподіл пам'яті додатка. Саме куча (через менеджер пам'яті) є стандартом для роботи з динамічною пам'яттю. Будь-яка куча є динамічною пам'яттю, але не будь-яка динамічна пам'ять – куча.

Також «кучою» називають не тільки сам механізм управління пам'яттю, але і регіон адресного простору, виділений для потреб кучі. Спочатку цей регіон малий або зовсім відсутній. У міру роботи програми (і виділення в ній пам'яті) спеціальний диспетчер, керівник кучами (менеджер пам'яті), розширюватиме цей регіон у міру необхідності або створюватиме нові регіони. А при звільненні блоків пам'яті в кучі менеджер пам'яті буде повертати системі відповідні сторінки фізичної пам'яті (в міру можливості).

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

Керована куча (Managed Heap). Помічена зеленим кольором. Керована куча є видом приватної пам'яті процесу, яка виділяється і використовується в .NET «складальником сміття» для більше не потрібних або пошкоджених даних.

Стек (Stack). Помічений оранжевим кольором. Стек – це структура даних, в якій доступ до елементів організований за принципом LIFO (last in – first out, «останнім прийшов – першим вийшов»). Операція переміщення елемента до стека називається «вштовхуванням» (push), а зворотна до неї – «виштовхуванням» (pop). Обидві операції можливі тільки по відношенню до верхівки стека (рис. 6.13).

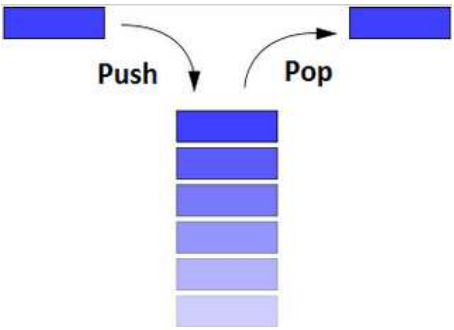


Рисунок 6.13 – Структура роботи Стеку

В адресному просторі будь-якого процесу є як мінімум одна спеціальна ділянка пам'яті, так звана «стек потоку». Кожен раз, коли в процесі створюється потік, система резервує ділянку адресного простору для стека потоку (у кожного потоку свій стек) і передає цій ділянці якийсь обсяг фізичної пам'яті. За умовчанням система резервує 1 Мб адресного простору і передає йому всього дві сторінки пам'яті. Стек вміє тільки розширюватися. Одного разу збільшивши свій розмір, він більш не зменшується.

VMMap дозволяє регулярно оновлювати карту розподілу пам'яті, що дозволяє скласти типові карти пам'яті для коректно функціонуючих додатків з тим, щоб надалі для полегшення діагностики можна було легко порівняти поточні кар-

ти з попередніми і виявити відмінності. Окрім цього, VMMap уміє працювати з сценаріями і дозволяє експортувати дані у формат .mmp (підтримується тільки цією утилітою). Ще одна дуже корисна можливість VMMap – опція Empty Working Set (Очистити робочу множину) в меню Refresh (Відновити): за допомогою цієї опції можна боротися з надмірним споживанням ресурсів пам'яті з боку «ненажерливих» додатків без завершення їх роботи або перезавантаження системи. Користуватися цією опцією слід з обережністю і лише у тому випадку, коли відомо напевно, яка поведінка після очищення пам'яті того або іншого процесу.

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 з вкладками та припускає декілька різних режимів подання інформації (рис. 6.14).

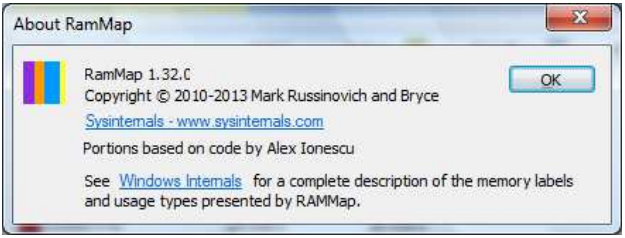


Рисунок 6.14 – RAMMap v.1.32c

За допомогою RAMMap можна швидко оцінити обсяги кешованих даних, що зберігаються в оперативній пам'яті, дізнатися, скільки пам'яті використовують окремі додатки, драйвери ядра і обладнання, а також отримати вичерпні відповіді на інші специфічні питання. Для зручності роботи передбачена можливість збереження і завантаження миттєвих знімків поточного стану оперативної пам'яті. RAMMap розповсюджується як у вигляді окремої програми, так і у складі пакета Sysinternals Suite.

***Wintest* (by Golubnychiy).** Даний програмний продукт створений на основі трьох демонстраційних програм, розроблених Джеффрі Ріхтером [3] (рис. 6.15).

Тому програма містить три основні пункти меню:

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

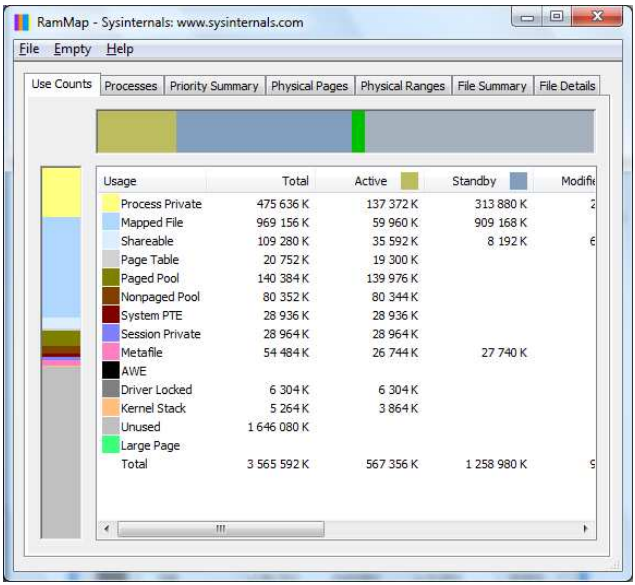


Рисунок 6.15 – Програмний продукт Wintest

1. VMStat. Ця програма виводить на екран вікно з результатами виклику функції *GlobalMemoryStatus*. Інформація у вікні оновлюється кожну секунду, так що VMStat цілком придатна для моніторингу пам’яті в системі (рис. 6.16).

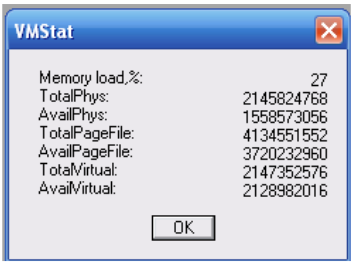


Рисунок 6.16 – Результати виклику функції *GlobalMemoryStatus*

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

Елемент *dwTotalPhys* (що показується як TotalPhys) відображає загальний обсяг фізичної (оперативної) пам'яті в байтах. На даній машині його значення становить 2 145 824 768, що на 1 658 880 байт менше 2 Гб. Причина, з якої *GlobalMemoryStatus* не повідомляє про повних 2 Гб, криється в тому, що система при завантаженні резервує невелику ділянку оперативної пам'яті, недоступну навіть ядру. Ця ділянка ніколи не скидається на диск. А елемент *dwAvailPhys* (що показується як AvailPhys) дає число байтів вільної фізичної пам'яті.

Елемент *dwTotalPageFile* (що показується як TotalPageFile) повідомляє про максимальну кількість байтів, яка може міститися в сторінковому файлі на жорсткому диску. Хоча VMStat показує, що поточний розмір сторінкового файлу становить 4 134 551 552 байти, система може варіювати його на свій розсуд Елемент *dwAvailPageFile* (що показується як AvailPageFile) підказує, що в даний момент 3 720 232 960 байт в сторінковому файлі вільні і можу бути передані будь-якому процесу.

Елемент *dwTotalVirtual* (що показується як TotalVirtual) відображає загальну кількість байтів, відведених під закритий адресний простір процесу. Значення 2 147 352 576 рівно на 128 Кб менше 2 Гб. Два розділи недоступного адресного простору – від 0x00000000 до 0x0000FFFF і від 0x7FFF0000 до 0x7FFFFFFF – якраз і складають цю різницю в 128 Кб.

dwAvailVirtual (що показується як AvailVirtual) – єдиний елемент структури, специфічний для конкретного процесу, який викликає *GlobalMemoryStatus* (решта елементів належить виключно до самої системи і не залежить від того, який саме процес викликає цю функцію). При підрахунку значення *dwAvaiWirtual* функція підсумовує розміри всіх вільних ділянок в адресному просторі процесу, що викликається. В даному випадку його значення говорить про те, що у розпорядженні програми VMStat є 2 128 982 016 байт вільного адресного простору. Віднявши із

значення *dwTotalVirtual* величину *dwAvailVirtual* отримуємо 18 370 560 байт – такий обсяг пам'яті VMStat зарезервувала в своєму віртуальному адресному просторі. Окремого елемента, який повідомляв би кількість фізичної пам'яті, використовуваної процесом у даний момент, не передбачено.

2. SysInfo. Ця програма вельми проста; вона викликає функцію *GetSystemInfo* і виводить на екран інформацію, повернену в структурі SYSTEM_INFO (рис. 6.17).

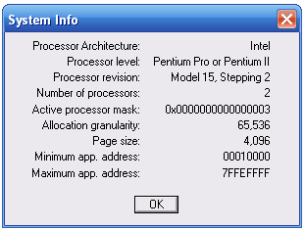


Рисунок 6.17 – Діалогове вікно з результатами виконання програми SysInfo

3. VMMap. Ця програма проглядає адресний простір довільного процесу і показує ділянки, що містяться в ньому, і блоки, присутні в ділянках (рис. 6.18).

Кожен елемент у списку – 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 область. Потім ці дані вносяться до списку, вказаного у таблиці 6.1.

Таблиця 6.1 – Список даних про ділянку

Базова адреса	Тип	Розмір	Блоки	Атрибути захисту	Опис
00000000	Free	65536			
...					

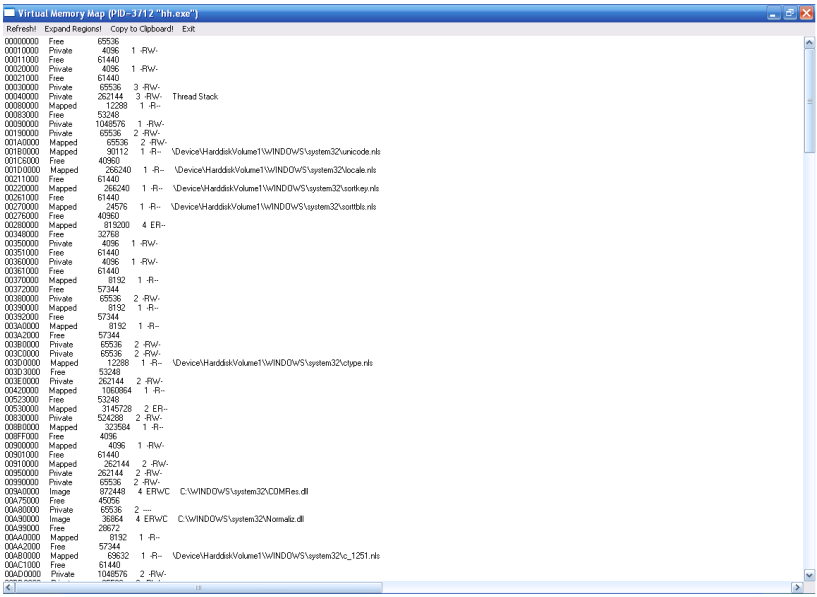


Рисунок 6.18 – робота програми VMMap

В основний цикл вкладений ще один цикл – він дозволяє отримувати інформацію про кожен блок поточної ділянки. На кожній ітерації з даного циклу викликається функція *ConstructBlklnfoLine*, що заповнює символьний буфер інформацією про блоки ділянки. Ці дані теж додаються до списку. За допомогою функції *VMQuery* проглядати віртуальний адресний простір процесу дуже легко.

В основі багатьох комп'ютерних архітектур розташовується 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 з процесів, що працюють на комп'ютері, в адреси фізичної оперативної пам'яті, реально встановленої на комп'ютері.

Завдяки MMU здійснюється підтримка багатьох надзвичайно ефективних і зручних технологій. Наприклад, якщо програма звертається до ділянки пам'яті з

адресою 0, механізм MMU може перенаправити це звернення за адресою 16384. Програма здійснює читання (або запис) байта за нульовою адресою, а насправді цей байт розташовується за адресою 16384. Інша програма може також звернутись за адресою 0, проте MMU здійснює перетворення адрес для кожної з програм по-різному. При зверненні до комірки з нульовою адресою другої програми він перенаправляє це звернення за абсолютно іншою адресою (наприклад, 32768). Іншими словами, завдяки MMU кожен процес, що працює в системі, думає, що володіє власним індивідуальним адресним простором від 0 до 4 Гбайт, хоча насправді дані, що належать цим процесам, зберігаються в різних областях одного і того ж фізичного адресного простору.

Якби MMU міг здійснити перенаправлення кожного байта ОП за іншою адресою, його таблиці відповідності програмних адрес фізично 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, вбудований в процесори типу Pentium, використовує сторінки розміром 4 Кбайт (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 відповідатиме елемент пам'яті з фізичною адресою 16385 і так далі. Проте елемент пам'яті з віртуальною адресою 4096 може відповідати абсолютно іншому, відмінному від 20480 (16384+4096) фізичному адресу, 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 і помітити її як відсутню.

Механізм відсутніх сторінок використовується також для організації *розріджених масивів (sparse arrays).* Уявіть, що ви працюєте з дуже великою матрицею дійсних чисел, для зберігання якої необхідно виділити 10 Мбайт оперативної пам'яті. Припустимо, що велика частина елементів матриці дорівнює нулю (або будь-якому іншому константному значенню). Матриці такого роду часто використовуються при вирішенні різного типу інженерних завдань.

У Windows вся матриця розділяється на сторінки по 4 Кбайти. Якщо сторінка містить нулі, вона позначається як відсутня. Коли програма, що працює з матрицею, звертається до такої сторінки, MMU повідомляє їй, що сторінка відсутня, і програма розуміє, що всі комірки сторінки дорівнюють нулю. Таким чином, на зберігання сторінок, цілком заповнених нулями, фізична пам'ять не витрачається.

ЗАГАЛЬНЕ ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Провести, якщо необхідно, завантаження системного (призначеного для користувача) процесу, вказаного в **стовпці 2** індивідуального завдання (табл. 6.2), дослідженню властивостей якого і буде присвячена лабораторна робота.

До закінчення роботи індивідуальний процес
НЕ ВИВАНТАЖУВАТИ з пам'яті

Таблиця 6.2 – Індивідуальні завдання для виконання

Вар.	Об'єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
1.	notepad.exe	Doubled access	читання	Створити додаток, за допомогою якого можна візуально досліджувати розподіл віртуальної пам'яті процесів. Як можливі процеси використовувати програми WINWORD, EXCEL і CALC. Надати користувачеві можливість вибору для дослідження необхідного процесу. Як статистичну інформацію слід використовувати відображення логічної адреси ділянок пам'яті, стани (вільне, готове до використання або

Продовження таблиці 6.2

Вар.	Об’єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
				зарезервоване), типи доступу, розміри блоків пам’яті, базові адреси, тип об’єктів
2.	shrpubw.exe	Doubled access	запис	Побудувати додаток, в якому під час запуску додатка створюється куча розміром в 100, 200, 500, 5000 байт. Завдання розміру проводити в окремому вікні. Потім пам’ять має бути перерозподілена і використовуватись як динамічний масив рядків (вміст задати самостійно, однотипні рядки не допускаються). Коли користувач вибирає один з пунктів меню, з кучі розподіляється пам’ять для зберігання нового рядка і до масиву додається указівник на цю пам’ять. Якщо в масиві не залишається місця, повинен проводиться запит на розширення масиву. Забезпечити **виведення в клієнтській зоні вікна вмісту рядків і значення указівника. Якщо** користувач обирає інший пункт меню, то звільняється останній розподілений рядок
3.	charmap.exe	Doubled access	копіювання	Створити додаток – розподілення сторінок віртуальної пам’яті. У діалоговому режимі забезпечити можливість виділення довільного блоку віртуальній пам’яті для процесу. За наслідками виділення вивести звіт, який повинен містити такі дані: − розмір (у байтах) зарезервований підділянку; − тип фізичної пам’яті, що використовується групою блоків даного регіону: MEM_FREE, MEM_IMAGE, MEM_MAPPED або MEM PRIVATE; − число блоків у вказаній ділянці; Передбачити додаткове виділення нових блоків, так само як і видалення вже виділених
4.	mmc.exe	Linear access	читання	Розробити додаток – моніторинг пам’яті в системі. Опціями додатка задати час оновлення даних вікна моніторингу. Вікно моніторингу повинне містити детальну

Продовження таблиці 6.2

Вар.	Об'єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
				інформацію і про фізичну, і про віртуальну пам'ять, а також дозволяти обробляти обсяг пам'яті, що перевищують 4 Гб. До такої інформації повинно належати: поточний стан використання пам'яті (число 0 ?100); загальний обсяг оперативної пам'яті; число байт доступної фізичної пам'яті; загальне число байтів файлу підкачки; загальне число байтів, доступних з файлу підкачки; загальне число байтів віртуальної пам'яті, доступної процесу, який викликається; число незарезервованих і невиділених байтів віртуальної пам'яті; число незарезервованих і невиділених байтів віртуальної пам'яті у розширеній частині віртуального адресного простору процесу, що викликається
5.	cliconfg.exe	Linear access	запис	Побудувати додаток, в якому повинен проводиться розподіл пам'яті для зберігання рядка з використанням функцій GloballAlloc() і GlobalReAlloc(). При створенні вікна додатка провести розподіл 27-символьного буфера (з нульовою ознакою кінця), в який розміщуються заголовні букви латинського алфавіту. При виборі користувачем додаткового пункту меню блок пам'яті перерозподіляється і в нього розміщуються ті ж букви, але в строковому зображенні. При кожному подальшому натисненні шляхом чергування повинен перерозподілятись буфер, а його вміст виводитись з нового рядка в клієнтській зоні вікна з вказівкою кількості перерозподілів
6.	colorcpl.exe	Linear access	копіювання	Під час запуску додатка створюється буфер розміром, заданим користувачем. Буфер призначений для зберігання *N* символів і нульової ознаки кінця. Надати можливість користувачеві ввести в буфер дані про прізвище, ім'я, по батькові розробника програми, назви факультету, номер групи і курсу. Якщо розміру буфера буде недостатньо,

Продовження таблиці 6.2

Вар.	Об'єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
				передбачити запит на його збільшення. Якщо первинний розмір буфера буде більший, то провести його зменшення. Таким чином, буфер має бути заповнений повністю.
7.	mobsync.exe	Doubled access	читання	Побудувати додаток – аналізатор параметрів віртуального адресного простору процесу, який повинен здійснювати виведення або в клієнтську зону вікна, або в діалогове вікно таку інформацію: архітектуру процесора; розмір сторінки; указівники 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.
8.	narrator.exe	Doubled access	запис	Розробити додаток, за допомогою якого користувач міг би вводити довільні адреси елементів віртуальної пам'яті і кількість байтів (Кбайт), а в робочій ділянці додатка виводився б їх вміст з розшифровкою призначення належності до певного модуля
9.	eudcedit.exe	Doubled access	копіювання	Розробити додаток, за допомогою якого користувач міг би вибрати довільний процес і проаналізувавши його віртуальний адресний простір, відобразити вміст тільки тєї частини, яка використовується різними системними *.dll. Відображення повинне показувати кількість блоків, атрибути захисту, належність до dll, тип (free, private, image або maped) і звичайно ж базові адреси
10.	sigverif.exe	Linear access	читання	Створити додаток, що проводить аналіз параметрів і вміст віртуальної пам'яті. Визначити відсоток поточного використання комп'ютером своїх ресурсів, обсягу фізичної пам'яті і файлу підкачки, обсягу вільного місця у файлі підкачки, обсягу вільного місця на диску. Крім того,

Продовження таблиці 6.2

Вар.	Об'єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
				використовуючи як управління іншими пунктами меню, надати можливість користувачеві встановлювати для довільних сторінок пам'яті прапорець GMEM_MOVEABLE
11.	wscript.exe	Linear access	запис	Створити додаток, що дозволяє аналізувати інформацію про доступну як фізичну, так і віртуальну пам'яті, при цьому передбачити можливість перевищення загального обсягу розміру в 4 Гбайт. Особливу увагу приділити аналізу параметрів файлу підкачки. Надати можливість користувачеві резервувати/звільняти, з числа вільного діапазону, довільний обсяг пам'яті
12.	iexpress.exe	Linear access	копіювання	Створити додаток, в якому проводиться резервування певного розміру пам'яті. Дослідити виникаючі типи помилок, залежно від типу доступу до пам'яті. Подати список усіх можливих помилок типу доступу. При виборі конкретної помилки проводити її моделювання. Забезпечити довідність і наочність, підтверджуючі наявність даної помилки
13.	magnify.exe	Doubled access	читання	Створити додаток, в якому під час запуску проводиться резервування віртуальної пам'яті заданого користувачем обсягу. Коли користувач вибирає один з пунктів меню додатка, то виділяються і використовуються 100 Кбайт віртуальної пам'яті. Значення розміщуються в кожному блоці пам'яті обсягом 1Кбайт. Доступ до всього виділеного блоку пам'яті змінюється на доступ «тільки для читання». Відбувається доступ до значення в блоці пам'яті і його відображення у вікні повідомлень. За допомогою іншого пункту меню користувач повинен зробити спробу змінити значення в пам'яті
14.	syskey.exe	Doubled access	запис	Після запуску додатку створюється три підпункти меню. За допомогою першого – запущеному процесу виділяється

Продовження таблиці 6.2

Вар.	Об’єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
				сторінка віртуальної пам’яті, заповнена двійковими одиницями. При виборі іншого – розкривається підменю, в якому користувач вручну встановлює типи доступу для однієї сторінки віртуальної пам’яті. При виборі третього – звільняється виділена сторінка віртуальної пам’яті. Передбачити можливість задання тільки одного типу доступу
15.	cleanmgr.exe	Double access	копіювання	Розробити додаток, що взаємодіє з системним кешем. Під час запуску додатка повинна створюватись куча, після чого пам’ять перерозподіляється і використовується у вигляді масиву рядків певного розміру. Коли користувач вибирає один пункт меню, з кучі розподіляється пам’ять для зберігання нового рядка і до масиву додається указівник на цю пам’ять. Передбачити відображення в клієнтській ділянці вміст вказівників і самих масивів. Якщо розмір кучі не достатній, необхідно вивести користувачеві повідомлення для того, щоб він міг іншим пунктом меню звільнити в зворотному порядку пам’ять, виділену рядку
16.	msinfo32.exe	Linear access	читання	При першому запуску додатку резервується буфер пам’яті. Розмір буфера налаштовується у відповідному діалоговому вікні. Буфер призначений для зберігання *N* символів і нульової ознаки кінця. У буфер автоматично поміщаються дані про розробника програми, номер групи та курсу, повністю заповнюючи буфер (передбачити відповідні обмеження). Ці відомості виводяться в клієнтську зону вікна. При виборі користувачем одного з пунктів меню (задати самостійно) блок пам’яті перерозподіляється, щоб звільнити місце ще для *K* символів. Ці додаткові символи записуються в буфер, відображають дату народження автора програми і відображаються в клієнтській зоні вікна. При виборі іншого пункту меню потрібно постійно очищати вміст клієнтської зони вікна

Продовження таблиці 6.2

Вар.	Об'єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
17.	msiexec.exe	Linear access	запис	Створити додаток, що проводить аналіз параметрів і вмісту віртуальної пам'яті. Визначити відсоток поточного використання комп'ютером своїх ресурсів, обсягу фізичної пам'яті і файлу підкачки, обсягу вільного місця у файлі підкачки, обсягу вільного місця на диску. Забезпечити можливість блокування / розблокування об'єктів пам'яті. Довести, що існують функції розблокування, що видаляють тільки дескриптор глобального об'єкта пам'яті, а також існують дескриптори, які додатково видаляють з пам'яті розблоковані сторінки
18.	mstsc.exe	Linear access	копіювання	Розробити додаток, що здійснює вибір довільного процесу, як призначеного для користувача, так і для системи. Відобразити в клієнтській ділянці вікна розподіл адресного простору цього процесу
19.	eventvwr.exe	Doubled access	читання	Побудувати додаток, що дозволяє зберігати рядок в блоці пам'яті глобальної кучі. Вміст рядка повинен містити найменування країн світу. При первинному запуску додатка повинен проводитись розподіл цього рядка у 27 байтному буфері. Після чого цей рядок відображається в 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 найменування країни задом наперед і відображаються в клієнтській зоні вікна, де вказати значення прапорців глобальної кучі: GHND; GMEM_FIXED; GMEM_MOVEABLE; GMEM_ZEROINIT; GPTR
20.	credwiz.exe	Doubled access	запис	Створити додаток, в якому проводиться резервування певного розміру пам'яті і призначення йому прав певного доступу. Досліджувати виникаючі типи помилок, залежно від типу доступу до пам'яті. Подати список усіх

Продовження таблиці 6.2

Вар.	Об'єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
				можливих помилок типу доступу. При виборі конкретної помилки проводиться її моделювання. Забезпечити докази і наочність, підтверджуючі наявність даної помилки
21.	cmstp.exe	Doubled access	копіювання	Розробити програму, яка намагається заповнити сторінку за сторінкою в клієнтській або системній ділянках віртуального адресного простору значенням, заданим користувачем. Користувач в опціях системи повинен мати можливість як вибору ділянках, задання діапазонів адрес, так і значення байта заповнювача. При спробі невдалого заповнення провести аналіз, наприклад, відсутність реальної сторінки за вказаною адресою або може бути заборонений до неї доступ. Результати мають бути виведені в клієнтській зоні вікна і збережені на диску
22.	osk.exe	Linear access	читання	Розробити додаток, що резервує віртуальну пам'ять обсягом в 1,5 Мбайти під час його запуску. При виборі другого пункту меню завдання провести виділення для використання блок 10 Кбайт віртуальної пам'яті. При цьому помістити в кожен 1 Кбайт блоку значення від 1 10, відповідно. Вивести в клієнтську ділянку результати. Надати можливість користувачеві встановлювати доступ по кожному з 1 Кбайт або тільки для читання, або читання і запис, або заборонити доступ. Надати можливість користувачеві самостійно задати нове значення будь-якого 1 Кбайта з виділеного блоку. Провести аналіз виникаючих помилок захисту пам'яті і відобразити їх у звіті
23.	verifier.exe	Linear access	запис	Розробити додаток, за допомогою якого досліджується механізм віртуальної пам'яті для управління масивом структур. Спочатку для масиву не резервується ділянка, і весь адресний простір вільний, що і відображено на карті пам'яті. Карта пам'яті має бути виділена: жовтим

Продовження таблиці 6.2

Вар.	Об'єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
				кольором – вільні ділянки; синім – зарезервовані; червоним – використані. За допомогою діалогового вікна здійснити резервування ділянки деякого обсягу (<100 Кбайт). Після цього вводиться індекс. За адресою, де розташований вказаний елемент масиву, передається фізична пам'ять. Далі карта пам'яті перерисовується і відображає стан ділянки, зарезервованої під весь масив. Будь-який елемент масиву, означений як зайнятий, можна звільнити кнопкою Clear
24.	psr.exe	Linear access	копіювання	Створити додаток, який за вибором користувача відповідних пунктів меню відображає в структурованому вигляді локальну таблицю дескрипторів пам'яті процесу. Відображення повинно включати детальний коментар призначення полів таблиць. Під час вибору одного з пунктів меню проводити додаткове виділення блоків віртуальній пам'яті заданих користувачем розмірів
25.	write.exe	Doubled access	читання	Побудувати додаток, в якому меню повинно містити два підпункти. При виборі першого – розкривається підменю, в якому користувач вручну встановлює для довільно взятої зі списку сторінки віртуальної пам'яті процесу один з таких станів: вільна, заповнена нулями, правильна (використовується активним процесом), змінена, запасна, погана. При виборі другого – звільняється виділена сторінка віртуальної пам'яті. Передбачити можливість задання довільних комбінацій типів доступу, а також прибрати взаємовиключні типи
26.	calc.exe	Doubled access	запис	За запуском додатка створюється три підпункти меню. За допомогою першого – запущеному процесу виділяється сторінка віртуальної пам'яті, заповнена двійковими одиницями. При виборі другого – розкривається підменю, в якому користувач міг би вручну встановлювати типи доступу для однієї

Продовження таблиці 6.2

Вар.	Об’єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
				сторінки віртуальної пам’яті. При виборі третього – звільняється виділена сторінка віртуальної пам’яті. Передбачити задання тільки одного типу доступу
27.	dxdiag.exe	Doubled access	копіювання	Створити додаток – розподілення сторінок віртуальної пам’яті. У діалоговому режимі забезпечити можливість виділення довільного блоку віртуальній пам’яті для процесу. За наслідками виділення вивести звіт, який повинен містити такі дані: ідентифікатор базової адреси ділянки віртуального адресного простору; атрибути захисту, присвоєного ділянці при його резервуванні. Передбачити додаткове виділення нових блоків, так само як і видалення вже виділених
28.	dialer.exe	Linear access	читання	Розробити додаток, що здійснює вибір довільного процесу, як призначеного для користувача, так і системного. У клієнтській зоні вікна провести відображення розподілу всього віртуального адресного простору з обов’язковою вказівкою таких параметрів: базова адреса ділянки; тип ділянки; розмір ділянки в байтах; кількість блоків у зарезервованій ділянці; атрибути захисту ділянки пам’яті; короткий опис вмісту поточної ділянки (повний шлях до файлу)
29.	mspaint.exe	Linear access	запис	Побудувати додаток, що моделює застосування функції GlobalDiscard(). Для цього при ініціалізації додатка провести розподіл пам’яті на декілька блоків однакового розміру, в кожен з яких помістивши назву факультетів університету. Вивести в діалогове вікно повний вміст вибраного користувачем блоку. Після цього застосувати у вказаному блоці GlobalDiscard(). Постійно здійснювати виведення дескрипторів глобального об’єкта пам’яті, поверненого функцією GloballAlloc(). Після чого провести повторний розподіл пам’яті функцією GloballReAlloc()

Закінчення таблиці 6.2

Вар.	Об'єкт дослідження	Тип тесту	Тип операції	Додаткове завдання
30.	MdSched.exe	Linear access	копіювання	Розробити додаток, що дозволяє в другому пункті свого меню вибирати системний процес, наприклад, lsass.exe або csrss.exe. Провести для нього дослідження віртуального адресного простору адрес, що відводиться процесу. Виведення статистичної інформації відображати у додаткове вікно. До такої статистичної інформації про адресний простір повинно виступати логічні і фізичні адреси ділянок пам'яті процесу, їх тип, статус, розмір, і інформація стосується використання системних файлів – динамічних бібліотек

ХІД РОБОТИ

Етап 1. Провести аналіз наявної фізичної пам'яті на комп'ютері. Для цього за допомогою Диспетчера пристроїв (Панель управління → Система або комбінація клавіш Windows+Pause Break) в меню View (Вигляд) виберіть команду Resources by Connection (Ресурси по підключенню) і розкрийте вузол Memory (Пам'ять). Результати занесіть до таблиці 6.3.

Таблиця 6.3 – Розподіл адрес фізичної пам'яті комп'ютера

Адреса		Розмір блоку, Кб	Призначення
початкова	кінцева		
0000 0000	0009F FFF	640	Системна плата
000A 0000	000BFFFF	768	Шина PCI
.	.	.	.
Всього, Кб:		1 408	

Етап 2. Провести аналіз використаної фізичної пам'яті і її завантаженість після завантаження операційної системи на комп'ютері. Для цього необхідно використовувати програму **FreeRAM** (BySoft). Результати, підкріплені скриншота-

ми, занести до таблиці 6.4.

Таблиця 6.4 – Аналіз фізичної пам’яті в FreeRAM (BySoft)

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

Етап 3. Провести аналіз використаної фізичної пам’яті і віртуальної пам’яті на комп’ютері, її завантаженість після завантаження операційної системи. Для цього необхідно використовувати програму **Mz RAM Booster**. Результати, підкріплені скриншотами, занести до таблиці 6.5.

Таблиця 6.5 – Аналіз пам’яті в Mz RAM Booster

Параметр	Значення параметра
Інформація про фізичну пам’ять	
Total Memory, Mb	
Used Memory, Mb	
Free Memory, Mb	
Інформація про віртуальну пам’ять	
Total Memory, Mb	
Free Memory, Mb	
Загальна інформація	
Total Mb Recovered, Mb	
Total Optimizations	

Етап 4.1. Провести аналіз фізичної пам’яті за допомогою монітора ресурсів для комп’ютера. Для цього необхідно використати програму **Resmon.exe,** вкладку *Пам’ять*. Результати фізичного розподілу пам’яті процесів помістити до таблиці 6.6. Подати в звіті гістограми розподілу фізичної пам’яті.

Таблиця 6.6 – Аналіз фізичної пам’яті в Resmon.exe

Параметр	Значення параметра
Зарезервований апаратний, Мб	
Використовується, Мб	
Змінено, Мб	
Зарезервовано, Мб	
Вільно, Мб	

Етап 4.2. Провести аналіз віртуальної пам’яті процесу, заданого в **стовпці 2** індивідуального завдання (див. табл. 6.2), за допомогою монітора ресурсів для комп’ютера. Для цього необхідно використати програму **Resmon.exe.** Результати розподілу віртуальної пам’яті процесів помістити до таблиці 6.7.

Таблиця 6.7 – Аналіз віртуальної пам’яті процесу __(найменування процесу)__ в Resmon.exe

Параметр	Значення параметра
Образ	
ІД процесу	
Помилок відсутності сторінки в пам’яті, сек.	
Завершено, Кб	
Робочий набір, Кб	
Загальний, Кб	
Приватний, Кб	

Етап 5.1. Провести аналіз загальних параметрів пам’яті ОС. Для цього необхідно використати програму **FreeRAM XP Pro.** Результати аналізу (меню *Файл > Звіт про стан пам’яті*) відобразити у вигляді таблиці 6.8.

Таблиця 6.8 – Звіт FreeRAM XP Pro Comprehensive Memory Report

Параметр	Значення
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:	
% 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:	
Cache Statistics	
Copy Read Hits %:	
Data Map Hits %:	
MDL Read Hits %:	

Закінчення табл. 6.8

Параметр	Значення
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 5.2. Провести аналіз параметрів процесу, заданого в **стовпці 2** індивідуального завдання (див. табл. 6.2). Для цього необхідно використати програму **FreeRAM XP Pro**. Результати аналізу (меню *Інструменти* → *Звіт про використання RAM)* – у вигляді таблиці 6.9.

Таблиця 6.9 – Розширений звіт про процеси у FreeRAM XP Pro

Параметр	Значення
Ім’я файлу процесу	
Обсяг пам’яті, що використовується	
Шлях до файлу	
Ідентифікатор процесу	
Кількість потоків	
Клас пріоритету процесу	
Ідентифікатор батьківського процесу	
Кількість помилок сторінок	
Максимальний робочий обсяг процесу	
Ємність використаної частини процесом файлу підкачки	
Максимальний обсяг використаного файлу підкачки	

Етап 6.1. Провести аналіз системної віртуальної пам’яті комп’ютера, використовуючи програму **CleanMem,** запустивши її графічну оболонку Mini_Monitor.exe. Відобразити скриншот. Використовуючи контекстне меню, занести дані до таблиці 6.10.

Таблиця 6.10 – Аналіз системної віртуальної пам’яті в CleanMem

Параметр	Значення	Опис
Current Size		
Peak Size		
Page Fault Count		
Minimum Working Set		
Maximum Working Set		
Transition Shared Pages		
Transition Shared Pages Peak		

Етап 6.2. Провести аналіз основних параметрів віртуальної пам’яті процесу, заданого в **стовпці 2** індивідуального завдання (див. табл. 6.2), за допомогою програми **CleanMem,** запустивши її графічну оболонку Mini_Monitor.exe. Використовуючи контекстне меню, занести дані до таблиці 6.11.

Таблиця 6.11 – Аналіз віртуальної пам’яті процесу __(найменування процесу)__ в CleanMem

Параметр	Значення	Опис
Caption		
Command Line		
Computer		
Creation Date		
Handle		
Handle Count		
Kernel Mode Time		
Max Working Size		

Закінчення табл. 6.11

Параметр	Значення	Опис
Min Working Size		
Name		
Parent Process		
Peak Page File Usage		
Peak Virtual Size		
Peak Working Set		
Priority		
Private Page Count		
Process ID		
Thread Count		
Virtual Size		
Windows Version		
Working Set Size (Memory Usage)		

Етап 7.1 Провести спостереження, оптимізацію та тестування пам’яті комп’ютера за допомогою програми **RAM Saver Pro**. Результати, підкріплені скриншотами, занести до таблиці 6.12.

Таблиця 6.12 – Дослідження пам’яті програмою **RAM Saver Pro**

Параметр	Значення	
	до оптимізації	після оптимізації
Усього RAM, Mb		
Вільно RAM, Mb		
Усього pagefile, Gb		
Вільно pagefile, Gb		

Етап 7.2 Провести тестування параметрів фізичної пам'яті комп'ютера в **стовпці 3, 4** індивідуального завдання (див. табл. 6.2). Результати дослідження швидкостей операцій занести до таблиці 6.13 і побудувати графік залежності від кількості потоків. Зробити висновок.

Таблиця 6.13 – Тестування пам'яті за допомогою програми **RAM Saver Pro**

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

Етап 8. Провести евристичне дослідження віртуального адресного простору процесу, заданого в **стовпці 2** індивідуального завдання (див. табл. 6.2), використовуючи програму **VMMap.exe.** Для цього після запуску програми в діалоговому вікні провести вибір необхідного процесу. Після чого визначити статистичну і адресну інформацію про розподіл віртуальних адрес, характеристику системних просторів процесу, кучі (heap), стеків і так далі. Як результат подати розподіл усього адресного простору у вигляді скриншоту (рис. 6.19). А також у вигляді таблиці 6.14 розподіл адрес.

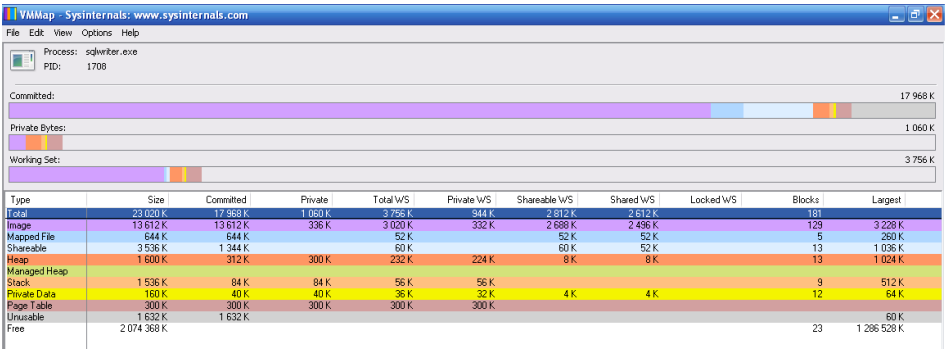


Рисунок 6.19 – Розподіл адресного простору

Таблиця 6.14 – Розподіл віртуального адресного простору процесу __(найменування процесу)__

Адреса	Тип блоку	Розмір	Захист	Опис
01000000	Image	..	Execute/Read	D:\...\..exe
..				
7FFFFFFF				

Етап 9. Провести аналіз розміщення образу файлу, заданого в **стовпці 2** індивідуального завдання (див. табл. 6.2), у фізичному і віртуальному адресних просторах. Для цього необхідно використати програму **RAMMap**. Результати занести до таблиці 6.15.

Таблиця 6.15 – Аналіз розміщення образу файлу у віртуальній пам'яті

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

Етап 10. Провести дослідження пам'яті, використовуючи програму **WinTest.exe** за допомогою трьох підпрограм: VMStat, SysInfo, VMMap. В останній програмі після запуску в діалоговому вікні провести вибір необхідного процесу. Після чого визначити статистичну і адресну інформацію про розподіл віртуальних адрес, характеристику системних просторів процесу. Як результат подати відомості про фізичну і віртуальну пам'ять у вигляді таблиці 6.16, розподіл усього адресного простору процесу – у вигляді таблиці 6.17.

Таблиця 6.16 – Характеристика пам’яті і системної інформації

Параметр	Значення	Параметр	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 6.17 – Розподіл адресного простору процесу __(найменування процесу)__

Базова адреса	Тип	Розмір	Блоки	Атрибути захисту	Опис
00000000	Free	65536			
...					

Після закінчення роботи потрібно очистити від прописаних параметрів автозапуск програм у гілці системного реєстру HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Додаткове завдання

Створити додаток, в якому 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 (див. табл. 6.2).** Третій пункт меню відображає інформацію про автора розробника програми з вказівкою факультету, номер навчальної групи і прізвище студента. Забезпечити підтримку довільного набору гарячих клавіш для кожного пункту меню.

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 проводиться взаємодія між оперативною пам’яттю і файлом підкачки?
4. Поясніть способи реєстрації зміни станів сторінок віртуальної пам’яті і механізми їх переміщення на диск і назад.
5. Як розподіляється системна віртуальна пам’ять?
6. Дайте визначення поняттю кучі. Які функції використовуються для роботи з нею?

Лабораторна робота 7

ДОСЛІДЖЕННЯ ВИКОНУВАНОГО ФАЙЛУ WINDOWS

Мета роботи: вивчити структуру виконуваних типів файлів в операційній системі Windows, провести аналіз структури файлового формату Portable Executable (PE), ознайомитись з розміщенням ресурсів у виконуваних файлах Windows, використовуючи програмні засоби сторонніх розробників, що дозволяють вносити корективи до параметрів різних типів ресурсів виконуваних файлів.

Указівки з підготовки до виконання лабораторної роботи

Необхідно вивчити структуру PE – рідного файлового формату Windows. Потрібно вивчити внутрішній порядок пристрою виконуваних файлів Windows. Додаткову інформацію при підготовці до роботи можна отримати з літератури:

1. Саймон Р. Windows 2003 API. Энциклопедия программиста. /Р.Саймон. пер. с англ. – Киев : ООО "ДиасофтЮП", 2004. – С. 1049–1066.

2. Румянцев П.В. Работа с файлами в Win32 / П. В. Румянцев – М.: Горячая линия. – Телеком, 2002. – С.92–157.

Теоретичні відомості

PE – це «рідний» файловий формат Win32. Його специфікації походять від Unix Coff (common object file format). «Portable executable» універсальний для платформи win32: завантажувач PE будь-якої win32-платформи розпізнає і використовує цей файловий формат, навіть коли Windows запускається не на PC CPU-платформі, хоча це не означає, що PE можна імпортувати 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. Тому знання цього формату дасть цінні пізнання внутрішньої структури Windows.

Важливим поняттям, про яке необхідно знати, є RVA (Relative Virtual Address – відносна віртуальна адреса). Багато полів в PE-файли задаються саме за допомогою їх RVA. RVA – це просто зсув даного елемента по відношенню до ад-

реси, з якої починається відображення файлу в пам'яті. 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 даної таблиці 1464h:

Virtual Address – Image Base = Relative Virtual Address

401464h – 400000h = 1464h

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

Для дослідження структури виконуваних файлів існують такі програмні засоби:

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. Найбільш функціонально насичена програма для перевірки внутрішньої роботи виконуваних файлів EXE, DLL, SYS, DRV, CPL, OCX, BPL, DPL, SCR та інші, для яких у користувача немає початкового коду (рис. 7.1).

PE Explorer дає можливість відкривати, переглядати і редагувати PE-файли, виконувати статичний аналіз, виявляти інформацію про функції виконуваного файлу та збирати 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 вибирається аналізований файл, PE Explorer перевіряє його і відображає у вигляді резюме інформацію про PE-заголовки і всі ресур-

си, що містяться у файлі PE. Звідси інструмент дозволяє досліджувати конкретні елементи у виконуваному файлі. Окрім того, PE Explorer включає PE Header Viewer, Exported/Imported API Function Viewer, API Function Syntax Lookup, Resource Viewer/Editor, Dependency Scanner & Disassembler.

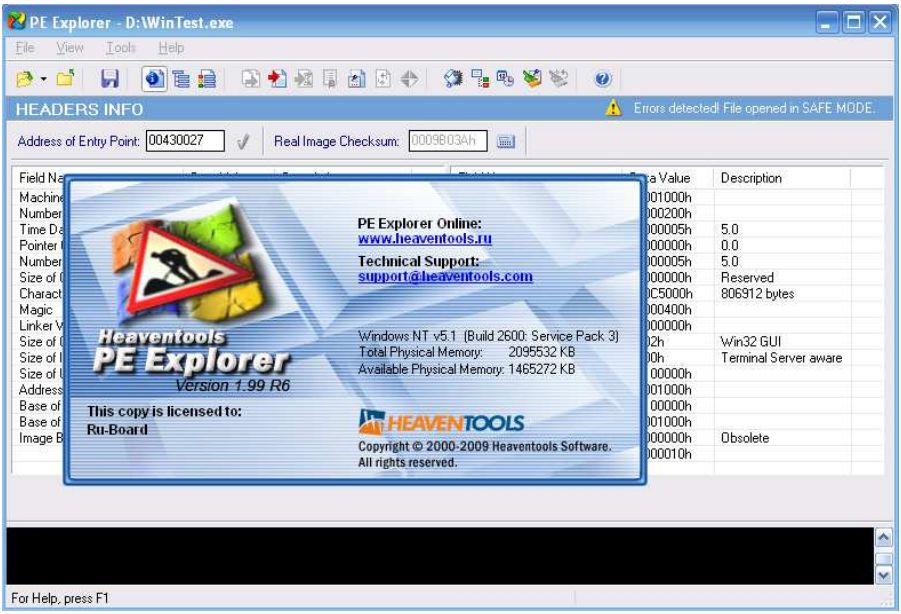


Рисунок 7.1 – Програмний продукт PE Explorer v1.99

PE Explorer призначений для використання в різних сценаріях, таких, як розробка програмного забезпечення, зворотний інжиніринг, зворотний аналіз безпеки і двійковий аудит процесів.

PEview version 0.9.9.0. Програму розроблено Вейном Дж. Редберном і розповсюджується абсолютно безкоштовно як спеціальне доповнення до програм аналізаторів PE виконуваних файлів. Програма написана на асемблері. Утиліта PEview забезпечує простий та швидкий спосіб перегляду структури і вмісту фай-

лів, записаних в PE- і COFF-форматах. Дозволяє переглядати файли типу EXE, DLL, SYS, OCX, CPL, SCR, OBJ, LIB, DBG і ін. (рис. 7.2).

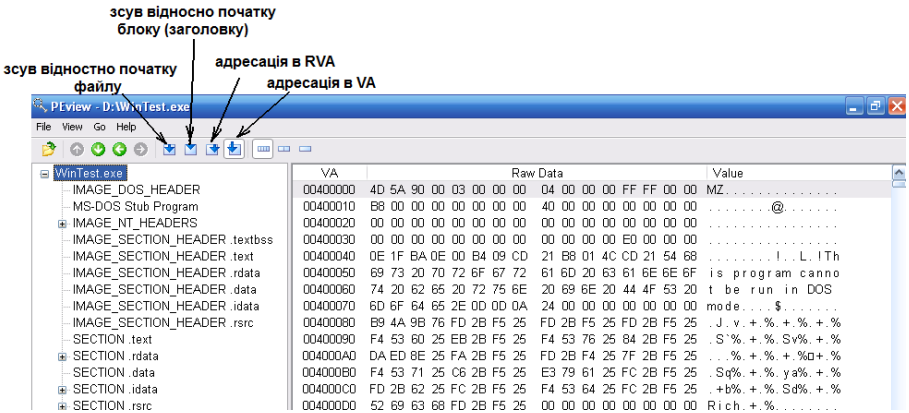


Рисунок 7.2 – Програма PEview version 0.9.9.0

PEiD v0.95 (PE iDentifier) – інструмент для дослідження PE файлів, що дозволяє дізнатися мову програмування, використану при написанні програми, назву пакувальника, за допомогою якого ця програма стиснута, або криптора, яким програму зашифрували (рис. 7.3).

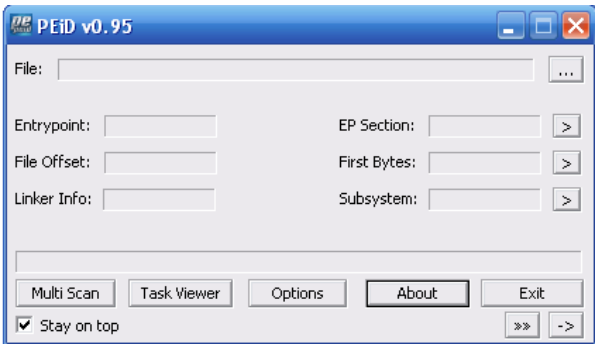


Рисунок 7.3 – інструмент для дослідження PE-файлів PEiD v0.95

Аналіз програмою PEiD проводиться за внутрішньою та зовнішньою базою сигнатур, поточна версія 0,95 може визначати **672** різних сигнатур в PE-файлах. У програмі PEiD є декілька рівнів сканування, можна обробляти цілі каталоги. Розширити функціонал PEiD можна за рахунок використання плагінів, які можна завантажити 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 він позначений, робоче вікно програми PEID завжди буде поверх інших вікон, якщо знятий, то поведінка вікна буде звичайною.

Зверху вікна мітка *File***:** текстове віконце і невелика кнопка – служить для вибору і відображення файлу, з яким працюватиме програма PEiD. Програма підтримує метод Drag-and-Drop. Після завантаження файлу програма починає аналіз.

У випадку, якщо PEiD нічого не знайдено, буде виведено "*Nothing found*". Якщо завантажений файл є не PE-файлом, програма видає "*Not a valid PE file*".

Аналіз PE-файлу здійснюється за допомогою сигнатур. Про сигнатури написано у файлі external.txt. Зовнішні сигнатури знаходяться у файлі userdb.txt.

Модифікація секції ресурсів виконуваного файлу може бути здійснена за допомогою додаткових програмних засобів.

Resource Hacker. Ресурси у виконуваних файлах (зокрема 32bit exe’s, dll’s, ocx’s і cpl’s) Windows розглядаються через вибір File → Open з меню Resource Hacker (рис. 7.4). Повний список ресурсів файлу буде показаний в деревоподібний структурі. Дерево ресурсу може бути повністю розширене або зруйноване при виборі Дерево View → Expand або Дерево View → Collapse відповідно з меню.

Спеціальним елементом ресурсу є тип ресурсу, його ім’я і ідентифікатор мови. Ресурси групуються в типи "ресурсів". Існує цілий ряд заздалегідь визначених типів (ікони, курсори, точкові рисунки, діалоги, меню, rcdata, і т.д) ресурсу, але програміст, можливо, визначив інші типи ресурсу.

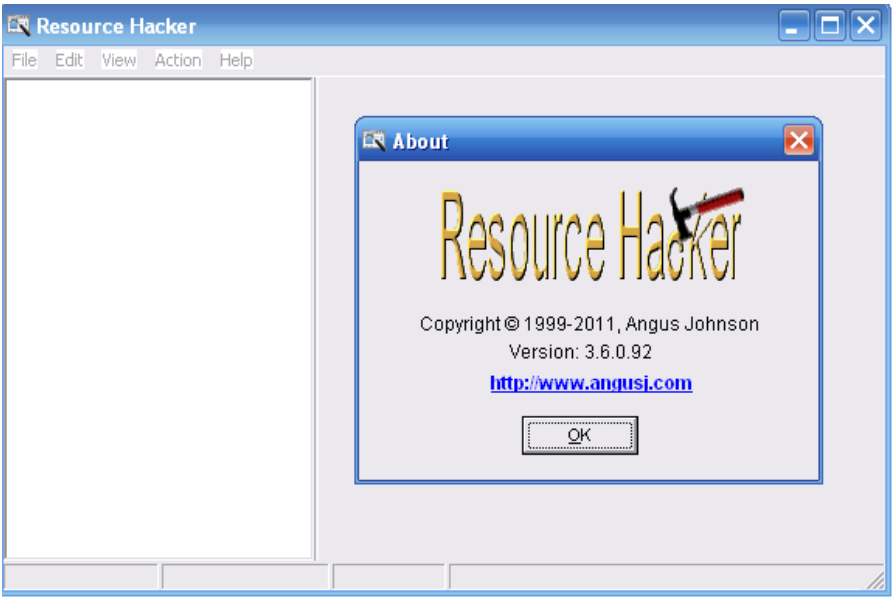


Рисунок 7.4 – Програма Resource Hacker

Елементи ресурсу збережені в межах їх відповідних типів ресурсу і мають «Ім'я ресурсу», який унікальний в межах того типу. Це «ім'я» може зазвичай бути або цілим значенням або буквено-цифровим значенням, проте деякі типи (таблиці рядків) ресурсу дозволяють тільки цілі значення у назві.

Кожен названий ресурс може мати більш ніж один певний елемент мови, щоб надати можливості програмам управляти багатьма 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, яка визначена Windows. Якщо елемент ресурсу – це "нейтральна мова", то значення буде нулем.

Multilizer – могутня за своїми можливостями програма для створення локалізацій додатків. Створена для перекладу меню програм й інших ресурсів програми іншими мовами (рис. 7.5).

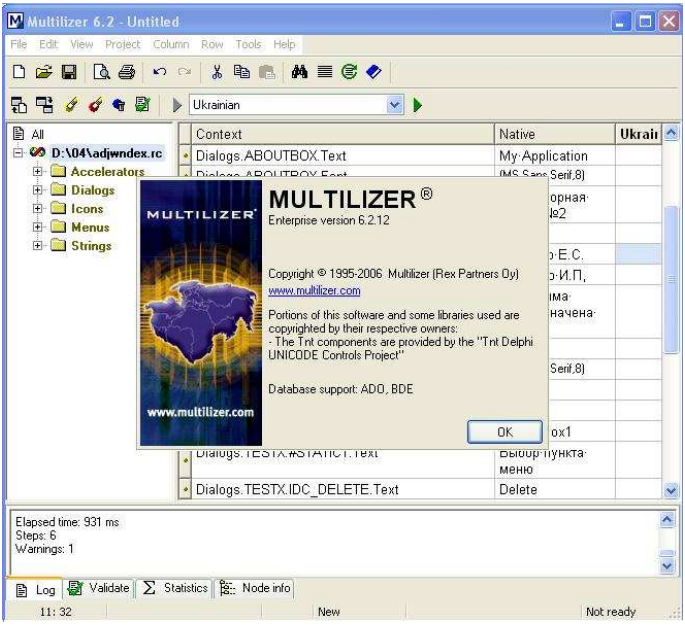


Рисунок 7.5 – Програма Multilizer

Multilizer підтримує велику кількість форматів, розрізняє тип мови програмування, на якій написана програма, дозволяючи редагувати програми, призначені для різних операційних систем. Програма підтримує більшість опцій, що дозволяють займатися локалізацією та є функцією імпорту вже готових перекладів.

LikeRusXP – 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. Для реалізації необхідно вказати, який файл треба перекласти, і програма автоматично зробить переклад (рис. 7.6).

Програма має вбудований візуальний редактор ресурсів з можливістю перекладу, сканер коду, кінцевий русифікатор має розмір від 50 Кб та інше. Має функцію самонавчання, заповнюючи пропуски в словнику, – інтелектуальний движок SmartLike.

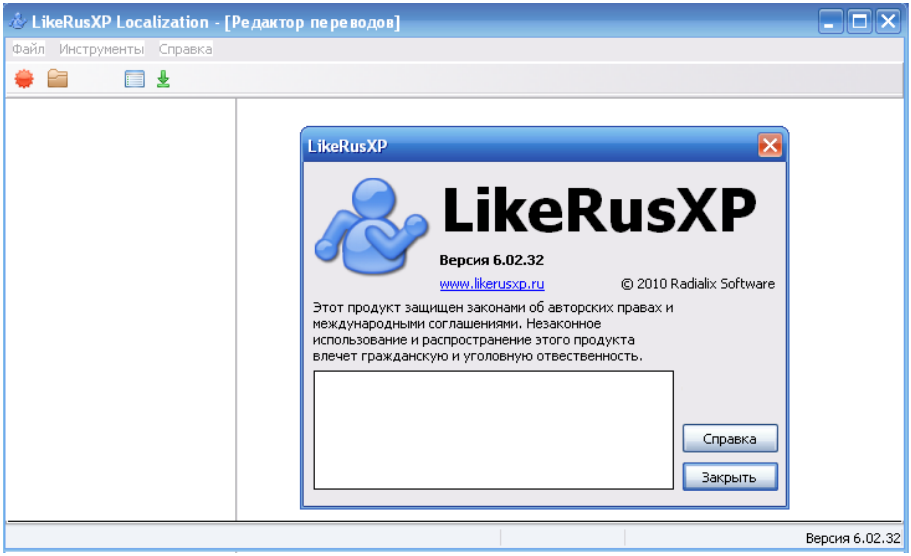


Рисунок 7.6 – Програма LikeRusXP

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 не перекладеного файлів.

ЗАГАЛЬНЕ ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

1. Аналітична частина

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

Провести дослідження DOS-заголовка виконуваних файлів, найменування яких

задане в індивідуальному завданні (табл. 7.1). Урахувати, що файли розташовуються в %SystemRoot%\System32. Результати дослідження занести до таблиці 7.2. Після таблиці 7.2 в звіті вказати вміст програми-заглушки DOS-заголовка.

Таблиця 7.1 – Індивідуальні завдання для виконання

Вар.	Об’єкт дослідження	Об’єкт дослідження	Файл 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

Закінчення табл. 7.1

Вар.	Об’єкт дослідження	Об’єкт дослі-дження	Файл для русифікації
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

Таблиця 7.2 – Порівняння DOS-заголовка виконуваних файлів

№№	RVA	Параметр	Значення	
			*.exe	*.dll
1	00000000	Сигнатура		
2	00000002	Довжина останньої неповної сторінки		
3	00000004	Довжина файлу в сторінках		
4	00000006	Число елементів в таблиці переміщення		
5	00000008	Довжина заголовка		
6	0000000A	Мінімум необхідної пам’яті за кінцем програми		
7	0000000C	Максимум необхідної пам’яті за кінцем 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		

Закінчення таблиці 7.2

№№	RVA	Параметр	Значення	
			*.exe	*.dll
13	00000018	Зсув першого елементу таблиці переміщення		
14	0000001A	Номер оверлею		
15	00000024	OEM ідентифікатор		
16	00000026	OEM інформація		
17	0000003C	Адреса Windows-заголовка		

* – замість символу (*) поставити найменування заданого файлу

Етап 2. Порівняльний аналіз структури Windows-заголовка exe і dll файлів. Провести дослідження Windows заголовків виконуваних файлів, найменування яких задане в індивідуальному завданні (табл. 7.1). Результати дослідження занести до таблиці 7.3.

Таблиця 7.3 – Порівняння Windows заголовка виконуваних файлів

№№	RVA	Параметр	Значення		Опис
			*.exe	*.dll	
1	..	Сигнатура			
Обов'язкова частина					
2	..	Процесор			
3	..	Число секцій			
4	..	Дата створення файлу			
5	..	Зсув COFF-таблиці			
6	..	Число байт COFF-таблиці			
7	..	Розмір IMAGE_OPTIONAL_HEADER			
8	..	Прапорці			
Необов'язкова частина					

Закінчення табл. 7.3

№№	RVA	Параметр	Значення		Опис
			*.exe	*.dll	
Стандартні поля					
9	..	Magic			
10	..	Версія лінкера			
11	..	Розмір коду			
12	..	Розмір даних, що ініціалізували			
13	..	Розмір неініціалізованих даних			
14	..	Адреса точки входу в програму			
15	..	RVA секції коду			
16	..	RVA секції даних			
Додаткові поля					
17	..	Базова адреса			
18	..	Межа вирівнювання секцій			
19	..	Розмір сектора на диску			
20	..	Версія ОС			
21	..	Версія виконуваного файлу			
22	..	Версія підсистеми ОС			
23	..	Розмір завантаженого модуля			
24	..	Розмір заголовків			
25	..	Контрольна сума			
26	..	Тип підсистеми інтерфейсу			
27	..	Прапорці завантаження Dll			
28	..	Розмір резервованого стека			
29	..	Розмір використовуваного стека			
30	..	Розмір резервованої кучі			
31	..	Розмір використовуваної кучі			
32	..	Кількість входів у масиві DataDirectory			

Етап 3. Порівняльний аналіз масиву секцій Windows-заголовка exe і dll файлів. Провести дослідження масиву секцій Windows-заголовка виконуваних

файлів, найменування яких задане в індивідуальному завданні (табл. 7.1). Результати дослідження занести до таблиці 7.4.

Таблиця 7.4 – Порівняння масиву секцій Windows-заголовка виконуваних файлів

№№	Імена елементів масиву	*.exe		*.dll	
		VA	Розмір	VA	Розмір
1	Секція експорту				
2	Секція імпорту				
3	Секція ресурсів				
4	Секція виключень				
5	Секція безпеки				
6	Секція налаштувань адрес				
7	Секція настройки				
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 exe і dll файлів

Провести дослідження таблиці секцій виконуваних файлів, найменування яких задане в індивідуальному завданні (табл. 7.1). Результати дослідження занести до таблиці 7.5.

Таблиця 7.5 – Таблиця секцій виконуваних файлів

№№	Ім’я	Virtual Size	RVA	Розмір секції	Зсув від початку файлу	Прапорці (характеристики)	Опис
Файл *.exe							
1	.text						
2	.data						
3	..						
Файл *.dll							
1	.text						
2	.data						
3	..						

Етап 5. Дослідження секції імпорту exe і dll файлів. Провести дослідження секції (.idata) виконуваних файлів, найменування яких задане в індивідуальному завданні (табл. 7.1). Результати дослідження занести до таблиці 7.6.

Таблиця 7.6 – Секція імпорту виконуваних файлів

№№	RVA таблиці імен імпорту	Час створення	ForwarderChain	Ім’я dll, що імпортується	RVA таблиці адрес імпорту
Файл *.exe					
1.				kernel32.dll	
2.				user32.dll	
3.				...	
Файл *.dll					
1.				kernel32.dll	
2.				user32.dll	
3.				...	

Примітка. Найменування бібліотек, що імпортуються, індивідуальні та кількість може перевищувати 3. Тоді в таблиці 7.6 відобразити дані про ВСІ бібліотеки.

2. Прикладна частина

Провести русифікацію вказаних в індивідуальному завданні (табл. 7.1) програм, послідовно використовуючи програми:

1. Resource Hacker.
2. Multilizer.
3. LikeRusXP.

Обов'язковими для модифікації є такі типи ресурсів: курсори, ікони, меню, діалогові вікна, таблиці рядків.

У звіті відобразити: скриншоти початкового стану ресурсних модулів, що підлягають модифікації; скриншоти кінцевого стану ресурсних модулів, що підлягають модифікації; порядок проведення модифікації.

Окремо подати **4** виконуваних файли: початковий і 3 модифікованих відповідною програмою.

3. Програмна частина – додаткове завдання

Програмним способом провести дослідження виконуваного тестового файлу, що скомпоновано компонувальником у двох версіях: RELEASE і DEBUG. Вивести у вікно ТІЛЬКИ відмінні секції і конструкції, однакові – не виводити.

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 файлу.
5. Яка специфіка таблиці об'єктів?
6. Призначення секції .idata.
7. Призначення секції .edata.
8. Призначення секції .data.
9. Призначення секції .text.
10. Призначення секції .rsrc.

Лабораторна робота 8

ДОСЛІДЖЕННЯ БІБЛІОТЕК ДИНАМІЧНОГО КОМПОНУВАННЯ

Мета роботи: ознайомитись з одним з найбільш важливих структурних елементів Windows – бібліотеками динамічного компонування (Dynamic Link Libraries, DLL), набути практичних навичок зі створення динамічних бібліотек на основі їх явного або неявного завантаження.

Указівки з підготовки до виконання лабораторної роботи

Необхідно вивчити принципи динамічного скріплення функцій і процедур Windows-додатків, звернути увагу на відмінність у використанні стандартних статичних бібліотек функцій мов програмування та динамічних бібліотек системного призначення.

Додаткову інформацію можна отримати з літератури:

1. Побегайло А. П. Системное программирование в Windows. / А. П. Побегайло – СПб.: БХВ-Петербург, 2006. – С.559–594.

2. Саймон Р. Windows 2000 API. Энциклопедия программиста. / Р. Саймон. пер. с англ. – Киев: ООО "ДиасофтЮП", 2002. – С.923–964.

3. Румянцев П.В. Азбука программирования в Win32 API. / П. В. Румянцев – М.: Горячая линия – Телеком, 2001. – С.258–269.

Теоретичні відомості

Бібліотеки динамічного компонування (DLL, або динамічні бібліотеки) є одним з найбільш важливих структурних елементів Windows. Більшість файлів, з яких складається Windows, є або програмні модулі, або модулі бібліотек, які динамічно підключаються. Велика частина принципів, які стосуються написання програм, цілком підходить і для написання цих бібліотек, проте є декілька важливих відмінностей.

Термін динамічне скріплення (dynamic linking) належить до процесів, які Windows використовує для того, щоб зв’язати функції в один з модулів з реаль-

ною функцією з модуля бібліотек.

Статичне скріплення (static linking) має місце в процесі створення програми, якщо для створення виконуваного файлу (.exe) зв'язуються воєдино всі об'єктні (.obj) файли і файли статичних бібліотек (.lib) і, як правило, скомпільовані файли опису ресурсів (.res). На відміну від цього, динамічне скріплення застосовється під час виконання програми.

Файли KERNEL32.DLL, USER32.DLL, GDI32.DLL, файли драйверів – бібліотеки, які динамічно підключаються. Ці бібліотеки можна використовувати у всіх програмах Windows.

Бібліотеки, які динамічно підключаються, можуть містити тільки ресурси або дані і не містити програм. Хоча модуль бібліотеки, яка динамічно підключається, може мати будь-яке розширення (наприклад, .exe .fon), стандартним розширенням, прийнятим у Windows, виступає .dll. Тільки ті бібліотеки, які динамічно підключаються, мають розширення .dll Windows може завантажити автоматично. Якщо файл має інше розширення, то програма повинна завантажити модуль бібліотеки явним чином, використовуючи функцію LoadLibrary (LoadLibraryEx).

Як правило, сенс динамічних бібліотек полягає у великому застосуванні. Бібліотеки використовують великий набір функцій, склад і зміст яких може змінюватись за час експлуатації і 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, які містяться всередині бібліотеки. Мізерна документація за форматом LIB-файлів називає LIB-файл архівами.

Усі LIB-файли починаються з однакової восьмої байтової сигнатури. Ця сигнатура визначена у файлі winnt.h:

```
#define IMAGE_ARCHIVE_START      "!<arch>\n"
```

Решта частини файлу являє 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 знаходяться на початку бібліотеки і існують для того, щоб компонувальник надалі міг швидко відшукати об'єктні файли в LIB-файлі. Початкові дані для члена архіву йдуть відразу за структурою IMAGE_ARCHIVE_MEMBER_HEADER, з якої починається кожний запис. Для більшості членів архіву записів початкові дані такі самі, як і в об'єктному файлі. Коли програма проводить виведення LIB-файлів, вона викликає ті ж процедури, що і при обробці об'єктного файлу. На рис. 8.1 показано формат LIB-файлів.

Сигнатура !<arch>.
IMAGE_ARCHIVE_MEMBER_HEADER (/Y)
Данные первого члена компоновщика
IMAGE_ARCHIVE_MEMBER_HEADER (/Y)
Данные второго члена компоновщика
IMAGE_ARCHIVE_MEMBER_HEADER (/Y)
Данные члена Longnames
IMAGE_ARCHIVE_MEMBER_HEADER (FOO.OBJ/)
Данные объектного файла
IMAGE_ARCHIVE_MEMBER_HEADER (/104)
Данные объектного файла
. . .

Рисунок 8.1 – COFF-формат LIB-файлів

Розглянемо 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 "/" являє ім’я члена. Якщо ім’я починається з символу "/", за яким йде десяткове число (наприклад, /104), то число є зсувом імені члена архіву всередині члена Longnames LIB-файлу. У попередньому прикладі ім’я члена починається з 104-го байта від початку ділянки 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 являє ім’я DLL, що містить функції, які імпортуються.

BYTE Date[12]

Дата і час створення члена – зберігається в десятковому ASCII-вигляді.

BYTE USERID[6]

Десяткове ASCII-подання ідентифікатора користувача – завжди NULL.

BYTE GROUPID[6]

Десяткове ASCII-подання ідентифікатора групи – завжди є рядком NULL.

BYTE Mode[8]

Десяткове ASCII-подання файлового режиму – завжди дорівнює нулю.

BYTE Size[10]

Розмір даних члена, поданий в десятковій ASCII-формі. Формат даних залежить від їх типу (вказаний у вже описаному полі Name).

BYTE EndHeader[2]

ASCII рядок \n.

Члени компонувальника. Всякий LIB-файл має дві секції членів компонувальника, що виконують функцію змісту для решти частини файлу. 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, по суті, переліки загальнодоступних символів у LIB-файлі разом із зсувами всередині файлу членів – об'єктних модулів, які містять загальнодоступні символи. Два члени компонувальника мають різні формати. Навіщо потрібно дві копії однакової інформації? Перший член компонувальника зберігає свою інформацію в тому порядку, в 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 і завжди використовує другий член.

Перший член компонувальника має такий формат.

DWORD NumberOfSymbols

Число загальнодоступних символів у даній бібліотеці. Це число подане у форматі big-endian (відображає спадщину COFF-формату 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 з цих членів – це член типу OBJ. Кожен елемент цього масиву відповідає імені символу в переліку подальших рядків ASCII.

BYTE StringTable[?]

Це нерозривна серія рядків стилів у пам'яті.

По суті, кожен елемент масиву Offset відповідає одному загальнодоступному символу, ім'я якого з'являється в ділянці StringTable. Наприклад, третій елемент масиву Offsets відповідає третьому рядку в ділянці StringTable:

```
First Linker Member:
 Symbols:  00000006
 MbrOffs   Name
 --------  -----------
 00000180  _DumpCAP@0
```

```
00000180  _StartCAP@0
00000180  _StopCAP@0 ...
```

Формат *другого члена компонувальника* заплутано через 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.

DWORD Offsets[NumberOfSymbols]

Масив файлових зсувів інших членів архіву. На відміну від першого члена компонувальника ці зсуви задані в природному форматі машини (тобто у форматі little-endian для i386).

DWORD NumberOfSymbols

Кількість загальнодоступних символів у масиві StringTable (отже, і кількість загальнодоступних символів у бібліотеці). Це поле до того ж містить кількість елементів в наступному далі масиві Indices.

WORD Indices[NumberOfSymbols]

Даний масив містить індекси (відлік починається з 1) масиву Offsets (описаний на два поля раніше). Цей масив йде паралельно рядкам масиву StringTable.

BYTE SiringTable[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 проглядає масив StringTable і обчислює відносний індекс рядка в масиві. Потім компонувальник використовує цей індекс для пошуку слова в масиві Indices. Нарешті, компонувальник віднімає 1 з цього слова в масиві Indeces 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 на-

бір рядків в стилі мови С, наступних одна за одною. Рядок розміщується у секції Longnames, якщо вона дуже велика, щоб укластися в 16 байт, зарезервованих для поля Name в структурі IMAGE_ARCHIVE_MEMBER_HEADER. В цьому випадку поле Name містить символ «/», за яким йде десяткове ASCII-подання зсуву рядка в секції Longnames.

Програма DLL2Lib.exe. Це сервісна програма перетворить DLL файл в його еквівалент статичного файлу бібліотеки. Після цього можна перемістити справжній файл DLL із статичним файлом бібліотеки, перебудувати цей додаток і поширювати з DLL файлом. Найважливіша особливість – це те, що процес обробки не вимагає яких-небудь джерел коду DLL файлів. Усі роботи походять з двійкового коду в двійковий. DLL to Lib інтегрується з багатьма утилітами, включаючи «Import Library Reference Information Generator», «Symbol Finder» і т.д. для того, щоб користувач був упевнений, що процес конвертування успішний (рис. 8.2).

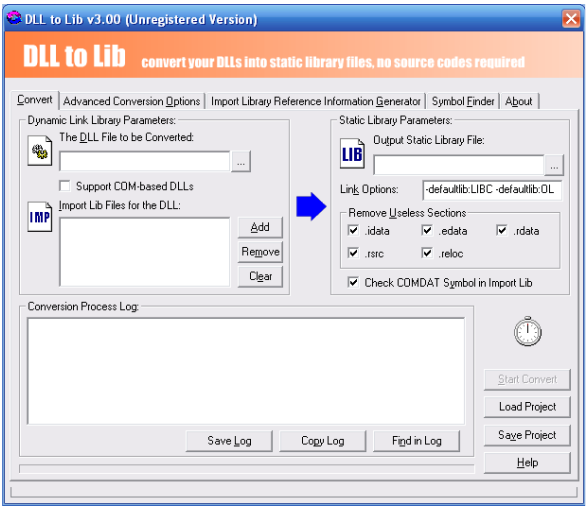


Рисунок 8.2 – Програма DLL2Lib.exe

Програма Processlnfo.exe. Ця програма демонструє, як створити дуже корисну утиліту на основі ToolHelp-функцій. Після запуску Processlnfo відкривається діалогове вікно.

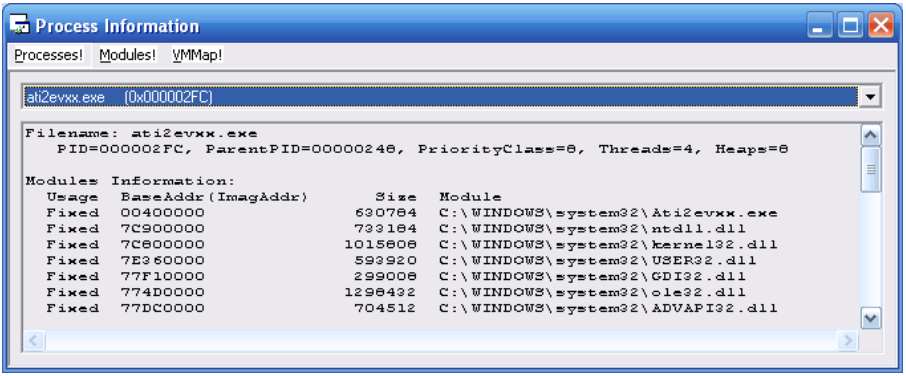


Рисунок 8.3 – Програма Processlnfo.exe

ProcessInfo спочатку перераховує всі процеси, що виконуються в системі, а потім виводить у верхній список, що розкривається, імена і ідентифікатори кожного процесу. Далі вибирається перший процес і інформація про нього показується у великому текстовому полі, доступному тільки для читання. Для поточного процесу повідомляється його ідентифікатор (разом з ідентифікатором батьківського процесу), клас пріоритету і кількість потоків, що виконуються зараз в контексті процесу.

В інформацію про модулі входить список усіх модулів (EXE- і DLL-файлів), спроектованих на адресний простір поточного процесу. Як фіксований модуль (fixed module) вважається той, який був неявно завантажений при ініціалізації процесу. Для явно завантажених DLL показуються лічильники числа користувачів цих DLL. У другому стовпці виводиться базова адреса пам'яті, на яку спроектований модуль. Якщо модуль розміщений не за заданою для нього базовою адресою, в дужках з'являється і ця адреса. У третьому стовпці повідомляється розмір модуля в байтах, а в останньому – повне (разом з шляхом) ім'я файлу цього модуля. І нарешті, внизу вказана інформація про потоки, що виконуються в даний момент у контексті поточного процесу. При цьому відображається ідентифікатор потоку (Thread ID, TID) і його пріоритет.

На додаток до інформації про процеси можна вибрати елемент меню Modules. Це змусить ProcessInfo перерахувати всі модулі, завантажені в системі, і помістити їх імена у верхній список. Далі ProcessInfo вибирає перший модуль і виводить інформацію про нього (рис. 8.4)

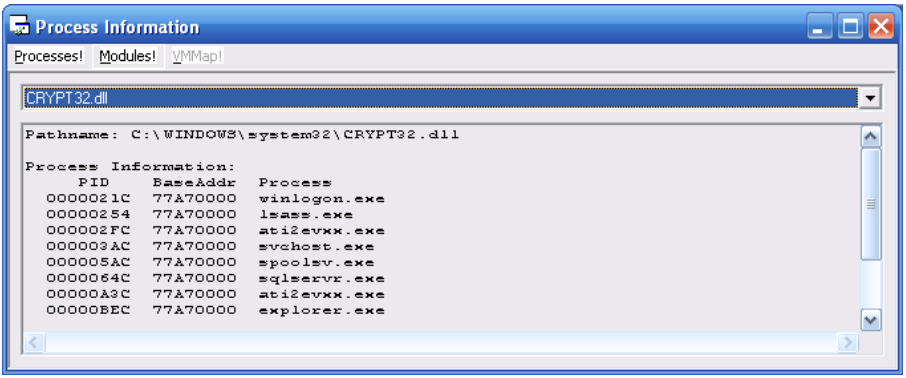


Рисунок 8.4 – Інформація про перший модуль

У цьому режимі утиліта ProcessInfo дозволяє легко визначити, в яких процесах задіяний даний модуль. Повне ім'я модуля з'являється у верхній частині текстового поля, а в розділі Process Information перераховуються всі процеси, що містять цей модуль. Там же показуються ідентифікатори і імена процесів, в які завантажений модуль, і його базові адреси в цих процесах.

Аналіз атак перехоплення DLL. Перехоплення DLL є можливим, оскільки практично всі додатки Windows в своїй роботі покладаються на бібліотеки DLL, як частину їх основної функціональності. DLL містять модульні компоненти коду, які розробники можуть використовувати всередині додатка для виконання різних функцій. Сама Windows покладається на такий же тип архітектури і 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.

Створені розробником DLL-файли пакуються і встановлюються разом з додатком. Проблема в тому, як додаток завантажує DLL. За умовчанням, коли для додатка не вказано статичний шлях до DLL, потрібний для цього додатка файл шукає динамічно. Виконуючи процедуру, додаток спочатку виконує пошук у каталозі, з якого виконувалось, потім виконує пошук у системному каталозі, 16-розрядному системному каталозі, каталозі Windows, у поточному каталозі, а потім у каталогах, перерахованих у змінному середовищі PATH ОС. Під час пошуку за цими шляхами додаток використовуватиме перший DLL, який знайдено.

Визначення уразливих додатків. Основною проблемою атаки перехоплення DLL є те, що компанія Microsoft не може випустити єдине виправлення для всіх уразливих додатків. Це неможливо тому, що призведе до некоректної роботи (або повної відмови) деяких додатків. Існує два способи визначення уразливості додатків. **Першим способом** є перевірка публічних ресурсів, викладених дослідниками в галузі безпеки, які вже виконали основну роботу. Одним з таких корисних ресурсів є ресурс http://www.exploit-db.com/dll-hijacking-vulnerable-applications/, що містить досить великий список (рис.8.5).

Другий спосіб вимагає більше роботи, але повинен виконуватися в середовищах з високим ступенем безпеки для пошуку уразливих додатків у системі. Цей набір називається **DllHijackAuditKitv2**, і його можна знайти http://blog.metasploit.com/2010/08/better-faster-stronger.html. Після завантаження необхідно зайти в систему з правами адміністратора, вилучити компоненти з ZIP файлу і виконати **01_StartAudit.bat.** Цей сценарій завантажить монітор Sysinternals Process Monitor і розпочне перевірку системи на уразливі додатки.

Завантаження монітора процесів частіше виявляється неуспішним, чим успішним, тому якщо відбувається дана ситуація при виконанні сценарію, можна вручну завантажити монітор Process Monitor. Після завантаження важливо переконатись, що Process Monitor розташований в тому ж каталозі, що і сценарії аудиту. Для запуску початкового сценарію аудиту необхідно час (рис. 8.6). Залежно від кількості додатків в ОС може вистачити від 15 хвилин до години.

Рисунок 8.5 – Список уразливих додатків Exploit DB

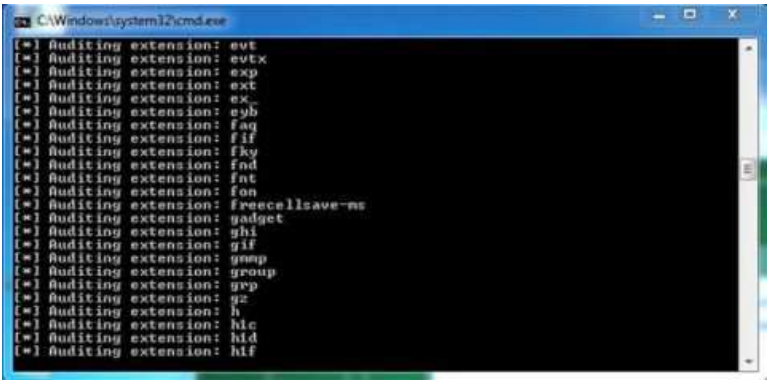


Рисунок 8.6 – Аудит додатків, пов’язаних з певними файлами

Після завершення роботи сценарію аудиту необхідно перейти на додаток монітора процесів, яким було ініційований сценарієм, і зберегти звіт, що згенеровано. Для цього натиснути Файл і обрати пункт Зберегти. Обов’язково зберегти файл у **форматі CSV** з ім’ям Logfile.CSV у каталозі набору аудиту.

Далі запускається **сценарій 02_Analyze.bat.** Цей сценарій проаналізує CSV файл і визначить потенційні уразливості. Якщо уразливість буде знайдено, додаток автоматично згенерує код засобу атаки, який можна використовувати для демонстрації уразливості (рис. 8.7).

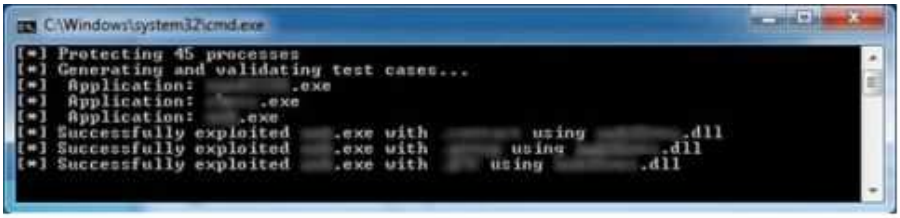


Рисунок 8.7 – Другий сценарій автоматично намагається використовувати виявлені уразливості

Після закінчення, інтерпретатор команд, залишений відкритим сценарієм, повинен 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 зловмисників 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 існування уразливості.

Наявність такої кількості вразливостей для перехоплення DLL становить цікавий випадок, оскільки його нелегко виправити за допомогою змін в операційній системі. Наявність властивостей впливає на велику кількість широко використовуваних додатків. Кращим захистом у цій ситуації є знання того, як працює уразливість, як перевіряти наявність уразливих додатків і як надавати правильну інформацію тим людям, які можуть виправити проблему.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Таблиця 8.1 – Індивідуальне завдання

Вар.	Файли для аналітичної частини				Файли для дослідницької частини	
	1	2	3	4	5	6
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
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

2. Створити динамічну бібліотеку, що містить функції перетворення рядка символів із прописних у рядкові.

3. Метод підключення DLL: використання явного завантаження DLL.

ВАРІАНТ 2

2. Створити динамічну бібліотеку математичних функцій: $y_1(X)=X_1 + X_2 + ... + X_n$ и $y_2(X)=X_1^1 + X_2^2 + ... + X_n^n$..

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 3

2. Створити динамічну бібліотеку функцій пошуку максимуму масиву цілих чисел. У функцію передається указівник на масив і число елементів масиву.

3. Метод підключення DLL: З використанням явного завантаження DLL.

ВАРІАНТ 4

2. Створити динамічну бібліотеку, що містить дві функції, які обчислюють довільний корінь зазначеного виразу.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 5

2. Створити динамічну бібліотеку, що містить функцію обчислення числа Фібоначчі.

3. Метод підключення DLL: використання явного завантаження DLL.

ВАРІАНТ 6

2. Створити динамічну бібліотеку, що містить функцію обчислення n-факторіал.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 7

2. Створити динамічну бібліотеку функцій обчислення суми довільного степеневого ряду, причому даній функції передається значення як кількості членів ряду, так і їхній степінь.

3. Метод підключення DLL: використання явного завантаження DLL.

ВАРІАНТ 8

2. Створити динамічну бібліотеку, що містить функції перетворення рядка символів з англійських букв у відповідні до російської розкладки клавіатури (наприклад, W → Ц; G → П і т.д.).

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 9

2. Створити динамічну бібліотеку математичних функцій обчислення квадратних або кубічних коренів виразу, степінь якого передається як параметр.

3. Метод підключення DLL: використання явного завантаження DLL.

ВАРІАНТ 10

2. Створити динамічну бібліотеку, що містить дві функції, які обчислюють cos(x) і sin(x).

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 11

2. Створити динамічну бібліотеку, що здійснює пошук у заданому текстовому фрагменті кількості заданої букви. Як параметр передається фрагмент тексту й букв, кількість яких у тексті потрібно знайти.

3. Метод підключення DLL: використання явного завантаження DLL.

ВАРІАНТ 12

2. Створити динамічну бібліотеку, що здійснює в одновимірному масиві пошук суми елементів масиву, розміщених до мінімального елемента.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 13

2. Створити динамічну бібліотеку, що здійснює у двовимірному масиві A операцію множення одного рядка на інший. Таким чином, як відповідь буде одновимірний масив C (наприклад, $c_1 = a_{11} * a_{21}$, $c_2 = a_{12} * a_{22}$, $c_3 = a_{13} * a_{23}$ і т.д.).

3. Метод підключення DLL: використання явного завантаження DLL.

ВАРІАНТ 14

2. Створити динамічну бібліотеку, що здійснює в одновимірному масиві пошук суми елементів масиву, розміщених між мінімальним і максимальним елементами.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 15

2. Створити динамічну бібліотеку, що здійснює в одновимірному масиві пошук суми елементів масиву з парними номерами.

3. Метод підключення DLL: використання явного завантаження DLL.

ВАРІАНТ 16

2. Створити динамічну бібліотеку функцій пошуку мінімуму масиву цілих чисел. Параметрами виступають указівник на масив і число елементів масиву.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 17

2. Створити динамічну бібліотеку, що здійснює в даній прямокутній матриці цілих чисел суму максимальних елементів у кожному рядку.

3. Метод підключення DLL: використання явного завантаження DLL.

ВАРІАНТ № 18

2. Створити динамічну бібліотеку, що здійснює в даній прямокутній матри-

ці цілих чисел суму елементів у тих рядках, що містять хоча б один негативний елемент.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 19

2. Створити динамічну бібліотеку, що здійснює для заданої матриці розміром 6*6 такі *k*, що *k*-ий рядок матриці збігається з *k*-м стовпцем, і суму елементів у тих рядках, які містять хоча б один негативний елемент.

3. Метод підключення DLL: використанням явного завантаження DLL.

ВАРІАНТ 20

2. Створити динамічну бібліотеку, що здійснює для заданої матриці перестановку стовпців, які відсортовані за зростанням елементів її останнього рядка.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 21

2. Створити динамічну бібліотеку, що здійснює в *n*-вимірному масиві пошук *n* максимальних елементів у кожному рядку.

3. Метод підключення DLL: використання явного завантаження DLL.

ВАРІАНТ 22

2. Створити динамічну бібліотеку, що здійснює пошук середнього арифметичного значення елементів одновимірного масиву.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 23

2. Створити динамічну бібліотеку, що здійснює пошук значення середньоквадратичного відхилення елементів одновимірного масиву.

3. Метод підключення DLL: використання явного компонування з DLL.

ВАРІАНТ 24

2. Створити динамічну бібліотеку математичних функцій:

$y_1(X)=1/X_1 + 1/X_2 + ... + 1/X_n$ и $y_2(X)=1/X_1^1 + 1/X_2^2 + ... + 1/X_n^n$.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 25

2. Створити динамічну бібліотеку, що містить функції перетворення рядка символів з рядкових у прописні.

3. Метод підключення DLL: використання явного компонування з DLL.

ВАРІАНТ 26

2. Створити динамічну бібліотеку, що здійснює сортування довільного масиву. Параметрами виступають указівник на масив і число елементів масиву.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 27

2. Створити динамічну бібліотеку, що містить дві функції, які обчислюють tg(x) і ctg(x).

3. Метод підключення DLL: використання явного компонування з DLL.

ВАРІАНТ 28

2. Створити динамічну бібліотеку, що здійснює в одновимірному масиві пошук суми елементів масиву з непарними номерами.

3. Метод підключення DLL: використання неявного компонування з DLL.

ВАРІАНТ 29

2. Створити динамічну бібліотеку, що містить функції перетворення рядка символів з російських букв у відповідні за розкладкою клавіатури англійські (наприклад, Ц → W; П → G і т.д.).

3. Метод підключення DLL: використання явного компонування з DLL.

ВАРІАНТ 30

2. Створити динамічну бібліотеку, що здійснює в одновимірному масиві пошук суми елементів масиву, розміщених між першими й останнім нульовими елементами.

3. Метод підключення DLL: використання неявного компонування з DLL.

1. Аналітична частина

Етап 1. Провести дослідження **6** динамічних бібліотек, розташованих в %SYSTEMROOT%\System32, вказаних в **п.1.** індивідуального завдання (табл. 8.1). У результаті дослідження подати у звіті заповнену таблицю 8.2 такого змісту (бібліотека A2Nusd.dll вибрана для прикладу).

Таблиця 8.2 – Характеристики динамічних бібліотек

№ п/п	Бібліотека	Кількість функцій, що експортуються	Кількість бібліотек, що імпортуються	Найменування 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
				Всього:	5 715 672
…	…	…	…	…	…
6					

Етап 2. Провести дослідження використання процесами **6** динамічних бібліотек, розташованих в %SYSTEMROOT%\System32, вказаних в **п.1.** індивідуального завдання за допомогою тестової програми **ProcessInfo**. В результаті дослі-

дження подати у звіті заповнену таблицю 8.3 такого змісту (бібліотека A2Nusd.dll вибрана для прикладу).

Таблиця 8.3 – Використання динамічних бібліотек в операційній системі

№ п/п	Найменування модуля	Інформація про бібліотеку (Process Information)		
		PID	BaseAddr	Process
1	A2Nusd.dll	00000A5C	7C630000	explorer.exe
		00000850	7C630000	TOTALCMD.EXE
…	…	…	…	…
6				

2. Дослідницька частина

Провести розробку в середовищі Visual Studio двох проектів створення динамічної бібліотеки, яка б містила функцію обчислення відповідно до **п.2** індивідуального завдання:

1. **DllCpp.dll** – з використанням засобів мови програмування C++ на Win32 API.

2. **DllCCs.dll** – з використанням засобів мови програмування C#.

При розробці проектів прагнути використовувати ОДНАКОВІ найменування змінних, які передаються в бібліотеку і повертаються з неї. Окрім вказаних бібліотек, для дослідження необхідно вибрати отримані при роботі компонувальника відповідні файли статичної бібліотеки (DllCpp.lib, DllCs.lib).

Етап 3. Дослідження бібліотечного (lib) файлу

Провести порівняльне дослідження структури lib файлів за допомогою програми **PEView.exe**, які були отримані в результаті компонування двох проектів: **DllCpp.lib** і **DllCCs.lib**. За необхідності скористатися програмою **DLL2Lib.exe** для отримання lib-файлу. Результати порівняння занести до таблиці 8.4.

Таблиця 8.4 – Дослідження lib-файлів

№	RVA	Параметр	Значення		Опис
			DllCpp.lib	DllCCs.lib	
1	..	IMAGE_ARCHIVE_START			
IMAGE_ARCHIVE_MEMBER_HEADER					
2	..	Name			
3	..	Date			
4	..	USERLD			
5	..	GROUPLD			
6	..	Mode			
7	..	Size			
8	..	EndHeader			
IMAGE_ARCHIVE_LINKER_MEMBER (Дані першого члена компонувальника)					
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 (Дані другого члена компонувальника)					
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 (Дані члена Longnames)					
27	..	Offset Table			
28	..	Symbol Table			

За результатами дослідження зробити висновки, основну увагу в яких приділити ступеню схожості (відмінності) розміщення функцій, що експортуються, змінних і т.д.

Етап 4. Дослідження динамічної бібліотеки (dll) файлу

Провести порівняльне дослідження структури dll файлів за допомогою програми **PEView.exe**, які були отримані в результаті компонування двох проектів: **DllCpp.dll** і **DllCCs.dll**. Результати порівняння занести до таблиць 8.5 і 8.6.

Таблиця 8.5 – Порівняння Windows заголовка динамічних бібліотек

№№	RVA	Параметр	Значення		Опис
			DllCpp.dll	DllCCs.dll	
1	..	Сигнатура			
Обов'язкова частина					
2	..	Процесор			
3	..	Число секцій			
4	..	Дата створення файлу			
5	..	Зсув COFF-таблиці			
6	..	Число байтів COFF-таблиці			
7	..	Розмір IMAGE_OPTIONAL_HEADER			
8	..	Прапорці			

Таблиця 8.6 – Таблиця секцій динамічних бібліотек

№№	Ім'я	Virtual Size	RVA	Розмір секції	Зсув від початку файлу	Прапорці (характеристики)	Опис
Файл DllCpp.dll							
1	.text						
2	.data						
3	..						
Файл DllCCs.dll							
4	.text						
5	.data						
6	..						

За результатами дослідження зробити висновки, основну увагу в яких приділити ступеню схожості (відмінності) розміщення функцій, що експортуються, змінних і так далі.

3. Прикладна частина

Етап 5. Проведення аналізу атак перехоплення DLL файлів

Визначити уразливі додатки в операційній системі за допомогою другого способу, використовуючи набір **DllHijackAuditKitv2** з програмою Process Monitor. Дане завдання необхідно виконувати з правами адміністратора. В результаті виконати 2 файли сценарію:

– **01_ StartAudit.bat** (подати викладачеві в звіті на перевірку файл Logfile.CSV, завантаживши його в Microsoft Excel).

– **02_ Analyze.bat** (подати викладачеві в звіті код засобу атаки на уразливості операційної системи).

4. Програмна частина – додаткове завдання

Створити виконуваний файл – додаток, в якому використовується створена в **п.2** індивідуального завдання динамічна бібліотека, яка реалізується **індивідуальним варіантом (п.3)** приєднання DLL до програми. Усередині DLL повинна містити набори функцій, що реалізовують це індивідуальне завдання.

Контрольні запитання

1. Для чого використовують бібліотеки?
2. Поясніть різницю між динамічним і статичним скріпленням.
3. У чому специфіка бібліотек, які динамічно підключаються?
4. У яких цілях використовується DLL, що розділяють пам’ять?
5. Чим відрізняється динамічне скріплення без імпорту?
6. Наведіть приклади системних DLL.
7. Опишіть процес створення DLL.

Лабораторна робота 9

ДОСЛІДЖЕННЯ СИСТЕМНОГО РЕЄСТРУ ОС WINDOWS

Мета роботи: вивчити структуру ключів реєстру, типи параметрів ключів, способи редагування реєстру; набути практичних навичок роботи з системним реєстром ОС Windows.

Указівки з підготовки до виконання лабораторної роботи

Необхідно вивчити порядок створення, видалення, запису і читання даних з використанням відповідно до власного варіанта тих або інших ключів реєстру, параметрів ключів, способів редагування реєстру, а також при підготовці до лабораторної роботи необхідно ознайомиться з теоретичним описом системного реєстру операційної системи Windows.

Додаткову інформацію можна отримати в таких джерелах:

1. Саймон Р. Windows 2003 API. Энциклопедия программиста: пер. с англ. /Р. Саймон. – К.: ООО "ДиасофтЮП", 2004. – С. 749–775.

Теоретичні відомості

Реєстр – це база даних, яка є частиною операційної системи Windows. Реєстр зберігає інформацію, яка використовується для ініціалізації і конфігурації додатків й управляє доступом до неї. Всі технології на платформі Windows інтенсивно використовують реєстр.

Основними групами відомостей, що зберігаються в реєстрі, є:

– **програми-установки.** 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 випадку інсталятор використовує реєстр для зберігання своїх налаштувань, дозволяючи правильно встановлювати і видаляти додатки, не чіпаючи спільно використовувані файли;

– **ядро ОС.** Зберігає багато відомостей в реєстрі про свою конфігурацію, у тому числі і дані про порядок завантаження драйверів пристроїв;

– **розпізнавач.** Під час кожного запуску комп'ютера програма NTDETECT.COM і ядро Windows розпізнає обладнання і зберігає інформацію в реєстрі;

– **диспетчер PnP (Plug and Play).** Позбавляє проблем з установлення нового обладнання. Зберігає інформацію в реєстрі;

– **драйвери пристроїв.** Зберігають свої параметри;

– **адміністративні засоби.** Наприклад, такі, як Панель управління, MMC (Microsoft Management Console) і ін.;

– **призначені для користувача профілі.** Це ціла група параметрів, унікальна для кожного користувача: налаштування графічної оболонки, мережевих з'єднань, програм і багато що інше;

– **апаратні профілі.** Дозволяють створювати декілька конфігурацій з 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 і документи відображаються в провіднику Windows Explorer. Частково це пов'язано з тим, що структура системного реєстру подібна до структури каталогів. Якщо вдаватись до термінології, що використовується в системному реєстрі, то еквівалентом папки або каталогу є ключ (key), а документа або файлу відповідає параметр з його значенням (value). Фактично це означає, що реєстр має ієрархічну структуру і складається з *розділів* і *пар параметр-значення*. На верхньому рівні ієрархії розташовані кореневі розділи (root keys) (табл. 9.1).

Кожен з вищеперелічених розділів містить у собі інші розділи – аналоги файлової системи. Registry має структуру дерева. Кінцевим елементом дерева реєстру є **ключі** або **параметри**. Ключі, які містяться в інших ключах, називаються **підключами** даного ключа. Всі ключі і параметри в межах підключа повинні мати унікальні імена.

Таблиця 9.1 – Кореневі розділи реєстру

№	Розділ (абревіатури)	Опис
1	HKEY_CLASSES_ROOT (HKCR)	Цей розділ містить визначення типів документів, зв'язків з файлами і інтерфейсу командного процесора
2	HKEY_CURRENT_USER (HKCU)	За допомогою цього ключа здійснюється доступ до призначених для користувача конфігурацій, програмного забезпечення
3	HKEY_LOCAL_MACHINE (HKLM)	Зберігає апаратні конфігурації, мережеві протоколи і класи програмного забезпечення
4	HKEY_USERS (HKU)	Використовується для зберігання вибраних користувачами глобальних параметрів, а також параметрів налаштування робочого столу
5	HKEY_CURRENT_CONFIG (HKCC)	Цей ключ установлює зв'язок з ключем відображення, що входить до складу підключа вибраної конфігурації config, HKLM, що міститься в ключі

HKEY_CLASSES_ROOT. Структура розділу декілька відрізняється від усіх останніх. Для кожного зареєстрованого розширення файлу є підключ (наприклад .bmp).

Значення цього ключа «За умовчанням» указує на підключ опису документа ("ACDC_BMP"), який розташований в тій же гілці основного розділу. У підключі опису документа і міститься ланцюжок ключів, що зберігають інформацію про асоціації, OLE, DDE.

HKEY_LOCAL_MACHINE. Інформація, збережена тут, використовується додатками, пристроями і 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 таблиці 9.2.

Таблиця 9.2 – Склад основного розділу HKEY_LOCAL_MACHINE

Розділ	Призначення
Hardware	Інформація про послідовні інтерфейси і модеми, які використовуються програмою HyperTerminal
SAM	Security Accounts Manager – адміністратор облікових даних у системі захисту (доступ до розділу блокований операційною системою)
Security	Інформація про те, який комп'ютер у мережі стежить за безпекою мережі і (чи допускає) підтримує даний комп'ютер віддалене управління
Software	Інформація про програмні засоби, встановлені на даному комп'ютері, і різні конфігураційні дані програм
System	Інформація даного розділу управляє запуском системи, завантаженням драйверів пристроїв, сервісом Windows і поведінкою системи

Параметри є у кожного ключа і підключа. Параметри складаються з трьох частин: ім'я параметра, тип параметра і його значення. Допустимі типи параметрів показані у таблиці 9.3. Кожному типу параметрів відповідає своя піктограма у вікні редактора реєстру.

Таблиця 9.3 – Типи параметрів системного реєстру

Тип параметрів	Опис
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

Закінчення табл. 9.3

Тип параметрів	Опис
	тип і відображаються в двійковому, шістнадцятковому або десятковому форматах
REG_SZ	Текстовий рядок фіксованої довжини
REG_EXPAND_SZ	Рядок змінної довжини. Цей тип даних включає змінні, що обробляються програмою або службою
REG_MULTI_SZ	Багаторядковий текст. Цей тип, як правило, мають списки і інші записи у форматі, зручному для читання. Записи розділяються 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_LINK	Символічне посилання Unicode. Тільки для внутрішнього використання (деякі кореневі розділи є таким посиланням на інші підрозділи)
REG_NONE	Параметр не має певного типу даних
REG_QWORD	64-розрядне число
REG_QWORD_-LITTLE_ENDIAN	64-розрядне число у форматі “гострокінцевий”. Еквівалент REG_QWORD
REG_RESOURCE_LIST	Список апаратних ресурсів. Використовується тільки в розділі HKLM\HARDWARE
REG_FULL_RESOURCE_-DESCRIPTOR	Дескриптор (описувач) апаратного ресурсу. Застосовується тільки в HKLM\HARDWARE
REG_RESOURCE_-REQUIREMENTS_LIST	Список необхідних апаратних ресурсів. Використовується тільки в HKLM\HARDWARE

При цьому типи параметрів REG_BINARY, REG_DWORD і REG_SZ є основними і найбільш часто використаними типами.

ЗБЕРІГАННЯ РЕЄСТРУ

Елементи реєстру зберігаються у вигляді атомарної структури. Реєстр розділяється 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 при кожному завантаженні, тобто є змінними. Під час запуску системи реєстр збирається з вуликів в єдину деревоподібному структуру з кореневими розділами. Перелік вуликів реєстру і їх місцеположення на диску наведено у таблиці 9.4.

Таблиця 9.4 – Вулики реєстру

Вулик	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_пользователя>	%USERPROFILE%\ntuser.dat
HKU\<SID_пользователя>_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, з такими розширеннями:

− ALT – резервна копія вулика HKLM\SYSTEM (відсутній в XP);

− LOG – журнал транзакцій, в якому реєструються всі зміни реєстру;

− SAV – копії вуликів в тому вигляді, в якому вони були після завершення текстової фази установлення.

Обмеження доступу користувачів до налаштування Windows здійснюється за допомогою зміни параметрів реєстру, що знаходяться в розділі HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\ ***. Параметри мають тип DWORD і можуть набувати значення:

"1" – обмеження увімкнене;

"0" – обмеження вимкнене.

Деякі параметри мають бути двійковими. Якщо необхідний такий параметр, то в таблиці на це буде вказано додатково (табл. 9.5–9.12).

Якщо деякі розділи, позначені зірочками, відсутні, то їх потрібно створити.

Таблиця 9.5 – Обмеження доступу до налаштувань екрана. ***=System

Параметр	Опис властивості параметра
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 9.6 – Обмеження доступу до налаштувань системи. ***=System

Параметр	Опис властивості параметра
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 9.7 – Обмеження доступу до налаштувань мережі. ***=Network

Параметр	Опис властивості параметра
NoNetSetup	Забороняє доступ до «Властивостей мережі» (з повідомленням)
NoNetSetupConfigPage	Забороняє доступ до «Властивостей мережі» (без повідомлення)
NoNetSetupIDPage	Недоступна вкладка «Ідентифікація» діалогу «Мережа»
NoNetSetupSecurityPage	Недоступна вкладка «Управління доступом» діалогу «Мережа»
NoEntireNetwork	Недоступна «Вся мережа» в «Мережевому оточенні»
NoFileSharingControl	Недоступне «розшарення» файлів і папок
NoPrintSharingControl	Недоступне «розшарення» принтерів
NoWorkgroupContents	Комп'ютери даної робочої групи в «Мережевому оточенні» не відображаються

Таблиця 9.8 – Обмеження доступу до налаштувань паролів. ***=System

Параметр	Опис властивості параметра
NoSecCPL	Ховає "Паролі" в "Панелі управління"
NoPwdPage	Ховає кнопку "Зміна паролів" у властивостях "Паролі" "Панелі управління"
NoAdminPage	Ховає вкладку "Видалене адміністрування" у властивостях "Паролі" "Панелі управління"
NoProfilePage	Ховає вкладку "Профілі користувачів" у властивостях "Паролі" "Панелі управління"

Таблиця 9.9 – Обмеження доступу до налаштувань принтерів. ***=Explorer

Параметр	Опис властивостей параметра
NoDeletePrinter	Забороняє видаляти принтер
NoAddPrinter	Забороняє додавати принтер
NoPrinterTabs	Забороняє доступ до "Властивостей принтера"

Таблиця 9.10 – Обмеження можливостей меню "Пуск" і "Робочого столу". ***=Explorer

Параметр	Опис властивості параметра
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"
NoRecentDocsMenu	Значення двійкового параметра 01 00 00 00 ховає "Документи" в меню "Пуск"
NoSetTaskbar	Відключає "Налаштування\Панель завдань" в 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

Параметр	Опис властивості параметра
NoTrayContextMenu	Забороняє контекстне меню "Панелі завдань"
NoViewContextMenu	Забороняє контекстне меню "Мого комп'ютера"
NoSetActiveDesktop	Видаляє "Робочий стіл Active Desktop" з меню "Налаштування"
NoSaveSettings	Не запам'ятовувати налаштування при виході з Windows
NoFolderOption	Видаляє "Властивості папки" з меню "Налаштування"
NoRecentDocsHistory	Не запам'ятовувати нещодавно відкритих документів
ClearRecentDocsOnExit	Очистити список нещодавно відкритих документів при виході

Таблиця 9.12 – Обмеження режиму MS-DOS. ***=WinOldApp

Параметр	Опис властивості параметра
NoRealMode	Заборона перезавантаження в режимі емуляції MS-DOS
Disabled	Заборона запуску сеансу MS-DOS

РЕДАКТОР РЕЄСТРУ

Редактор реєстру (regedit.exe) – основний інструмент користувача для маніпуляції з реєстром, який надається Microsoft.

Для запуску редактора реєстру використовується меню Пуск → Виконати. У вікні імені файлу необхідно ввести regedit.exe і натиснути ОК (рис.9.1). Редактор реєстру дає можливість переглядати вміст реєстру, створювати і видаляти підрозділи і пари *параметр = значення*, виконувати експорт-імпорт усього реєстру або його частини.

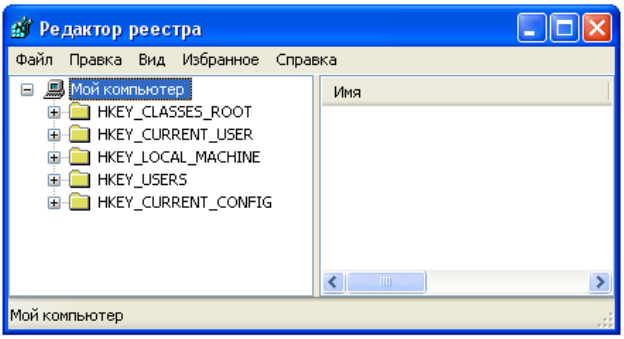


Рисунок 9.1 – Головне вікно

Reg-файл – це файл, що має певну структуру і містить інформацію, яка може бути імпортована в реєстр. Якщо була заблокована робота з редактором реєстру, то найбільш легким способом редагувати реєстр буде створення і імпортування reg-файлу.

При побудові reg-файлу можна скористатись блокнотом. На першому рядку повинні розміщуватися назва і версія редактора реєстру:Windows Registry Editor Version 5.00. Після чого обов'язково залишають один порожній рядок. Далі вказують розділ реєстру, в якому треба прописати або змінити якісь параметри. Назва розділу має бути поміщена в квадратні дужки [...]. Нижче прописуються параметри, які треба додати – один параметр у рядок. Якщо треба провести зміни в декількох розділах, то необхідно залишити 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 створений такий файл, необхідно просто запустити його як звичайну програму. Буде виданий запит про необхідність провести зміни в реєстрі, і після позитивної відповіді інформація з файлу буде імпортована. Про результати імпортування Windows повідомить у вікні, що з'явилося після цього.

Додавання параметрів. В наведеному вище прикладі додаються параметри за допомогою рядків типу "param1"="znachenie1". Тобто таким чином додається СТРОКОВИЙ параметр з ім'ям "param1" і значенням "znachenie1". Але ж існують ще і параметри двійкові і DWORD. Формат запису для їх додавання дещо інший. Для параметрів типу DWORD використовується рядок

```
"param"=dword:XXXXXXXX
```

Тут "param" – ім’я параметра, dword – вказує на тип цього параметра (букви мають бути обов’язково маленькі!) і після двокрапки йде значення з восьми цифр в шістнадцятковому (!) форматі. Проте більшість параметрів DWORD мають значення або 0, або 1, тому необхідно написати відповідно або 00000000, або 00000001 замість значків XXXXXXXX. Пропуски в рядку не допускаються.

Для додавання двійкового параметра формат запису декілька інший:

"param"=hex:XX,XX,XX.... ,

де "param" – назва параметра; hex –указівник на двійковий параметр; XX,XX,XX – шістнадцяткові числа, відокремлені комою. Наприклад, якщо треба додати двійковий параметр рівний "be 00 00 00", то пишеться рядок

"param"=hex:be,00,00,00

У реєстрі існують параметри "За умовчанням" ("Default"). Щоб присвоїти їм якесь значення через reg-файл, треба додати рядок:

@="znachenie",

де @ – вказує на присвоєння значення параметра "За умовчанням", не виділяється лапками.

За допомогою reg-файлів можна не тільки встановлювати нові параметри, але і видаляти їх. Наприклад, для видалення розділу з реєстру треба перед його ім’ям у квадратних дужках поставити символ "-":

[-HKEY_LOCAL_MACHINE\Software\QuickSoft\QuickStart]

Для видалення окремих параметрів використовується синтаксис:

"param"= -

Редактор реєстру підтримує тільки основні типи даних параметрів реєстру. Для виконання операцій з даними інших типів необхідно використовувати функції Win32 API.

ФУНКЦІЇ WIN32 ДЛЯ РОБОТИ З РЕЄСТРОМ

У таблиці 9.13 зведені всі функції системного реєстру, а опис основних йде безпосередньо після таблиці.

Таблиця 9.13 – Зведення функцій роботи з системним реєстром

Функція	Призначення
RegCloseKey	Закриває відкритий ключ системного реєстру
RegConnectRegistry	Виконує з'єднання з зумовленим дескриптором системного реєстру на іншому комп'ютері
RegCreateKeyEx	Створює новий підключ
RegDeleteKey	Видаляє ключ з системного реєстру
RegDeleteValue	Видаляє значення з системного реєстру
RegEnumKeyEx	Перераховує всі підключі даного ключа
RegEnumValue	Перераховує всі значення даного ключа
RegFlushKey	Одразу записує всі зміни, вироблені в системному реєстрі
RegLoadKey	Завантажує розділ у кореневий ключ, що знаходиться на вершині ієрархії
RegNotifyChangeKeyValue	Указує на момент зміни ключа або значення в системному реєстрі
RegOpenCurrentUser	Відкриває ключ HKCU для користувача поточного потоку
RegOpenKeyEx	Відкриває існуючий 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 (з розширеними типами даних Win32)
RegReplaceKey	Замінює ключ вмістом файлу під час перезапуску
RegRestoreKey	Зчитує вміст розділу в раніше збережений ключ
RegSaveKey	Зберігає значення і підключі даного ключа у файлі вулика
RegSetValueEx	Присвоює ключу значення (з новими типами даних)
RegUnLoadKey	Видаляє розділ з системного реєстру

ПРОГРАМИ ДЛЯ РОБОТИ З РЕЄСТРОМ

До програм сторонніх виробників для роботи з реєстром належать:

6 *AccessEnum* (by Bryce Cogswell).

7 *AutoRuns* (by Mark Russinovich).

8 *Process Monitor* (by Mark Russinovich).

9 *RegShot (freeware).*

10 *Regcleaner* (by Jouni Vuorio).

AccessEnum. Утиліта, що показує призначені права певним папкам або ключам реєстру (рис. 9.2). Застосування даної програми дозволить визначити незахищені місця в системі і можливі напрями атак на неї.

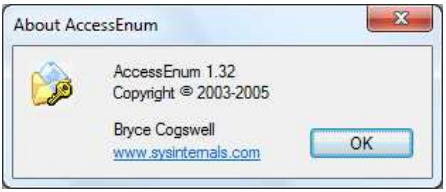


Рисунок 9.2 – Утиліта ***AccessEnum***

Гнучка модель безпеки операційних систем Windows на основі технології NT дозволяє повною мірою управляти безпекою і правами доступу до файлів. Проте розподіл прав є таким, щоб усі користувачі мали належний доступ до файлів, каталогів і розділів реєстру, що може виявитись складним завданням. *Вбудованого в систему способу швидкого перегляду прав доступу користувачів до дерева каталогів або розділів реєстру не існує.*

Програма AccessEnum створює детальний звіт про параметри безпеки файлової системи і реєстру, що робить її ідеальним інструментом для пошуку проломів у системі безпеки і визначення зайвих прав доступу, які слід знизити.

Утиліта AccessEnum проводить сканування всіх файлів у вибраній папці або ключів у розділі реєстру і виводить результат у вигляді, зручному для перегляду користувача (рис. 9.3).

Рисунок 9.3 – Вигляд результату сканування Утиліта AccessEnum

Інтерфейс програми простий, робота не ускладнена відсутністю русифікації. Після вибору папки або ключа реєстру потрібно натиснути кнопку Scan. Отриману таблицю результатів завжди можна впорядковувати за іменем об'єкта, дозволеними або забороненими правами.

AutoRuns. Перевіряє більшу кількість місць реєстру, з яких походить автозапуск програм, чим будь-який інший монітор автозавантаження. Показує, які програми налаштовані на запуск у процесі завантаження або входу в систему, програми відображаються в тому порядку, в якому система Windows обробляє їх.

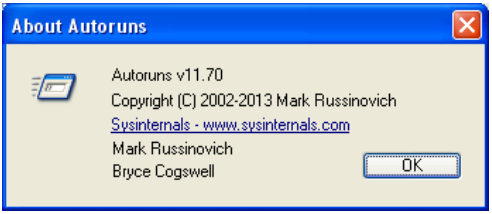


Рисунок 9.4 – Утиліта AutoRuns

Після запуску Autoruns показує, які застосування налаштовані на автомати-

чний запуск, а також надає повний список розділів реєстру і каталогів файлової системи, які можуть використовуватись для задання автоматичного запуску.

Елементи, які показує Autoruns, належать до декількох категорій (рис. 9.5):

– об'єкти, що автоматично запускаються при вході в систему;

– додаткові компоненти провідника;

– додаткові компоненти Internet Explorer (включаючи 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.

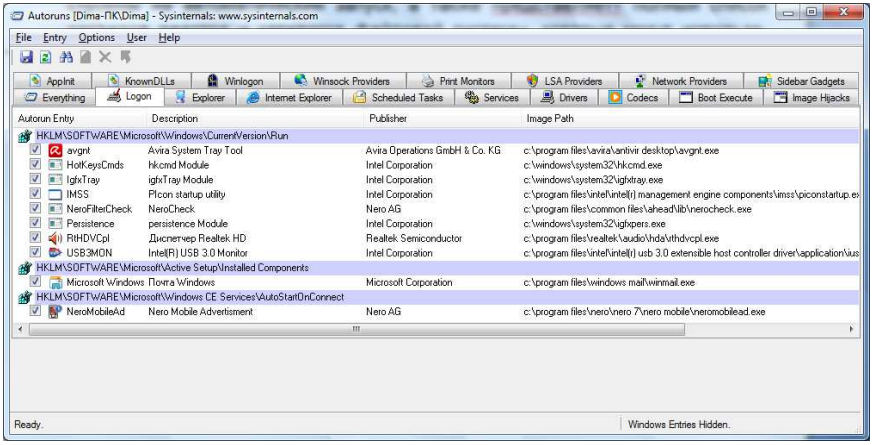


Рисунок 9.5 – Елементи Autoruns

Щоб проглянути об'єкти необхідної категорії, що автоматично запускаються, досить вибрати потрібну вкладку.

Щоб перейти до розділу реєстру або каталогу файлової системи, який відображається у вікні програми, або до налаштування об'єкта, що запускається автоматично, досить виділити потрібний елемент і скористатись командою меню

або кнопкою панелі інструментів **Jump** (Перейти).

Щоб відключити 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 проглянути елементи, що автоматично запускаються, для облікових записів інших користувачів – пункт меню **User** (Користувач).

Process Monitor – є вдосконаленим інструментом відстежування процесів для Windows, який в режимі реального часу відображає активність файлової системи, реєстру, а також процесів і потоків.

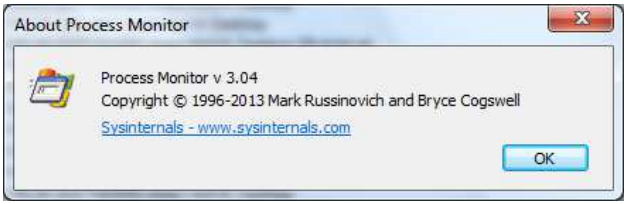


Рисунок 9.6 – Інструмент Process Monitor

У програмі поєднуються такі можливості:

- відстеження запуску і завершення роботи процесів і потоків, включаючи інформацію про код завершення;
- відстеження завантаження образів (бібліотек DLL і драйверів пристроїв, що працюють у режимі ядра);
- більше даних про параметри операцій введення і виведення;
- фільтри, які не призводять до втрати даних;
- збір стеків потоків для кожної операції дозволяє в більшості випадків визначити початкову причину виконання операції;
- достовірний збір інформації про процеси, включаючи шлях до образу процесу, командний рядок, а також ID-користувача і сесії;
- колонки, що налаштовуються і переміщуються, для властивості події;

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

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

– дерево процесів відображає взаємодії між усіма процесами, перерахованими у відомостях трасування;

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

– підказки до процесів для перегляду інформації про образ процесу;

– детальні підказки дозволяють дістати зручний доступ до форматів даних, які не вміщуються в колонці;

– пошук, що припиняється;

– запис у журнал усіх операцій під час завантаження системи.

Найпростіше ознайомитися з можливостями програми Process Monitor, прочитавши файл довідки і спробувавши скористатися кожним пунктом меню і налаштуванням програми на робочій системі (рис. 9.7).

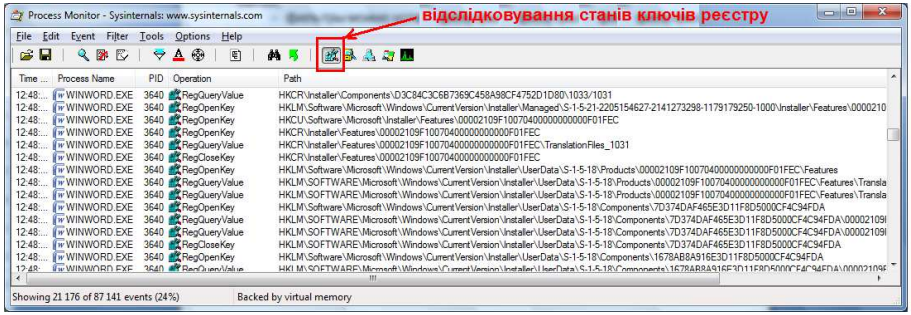


Рисунок 9.7 – Результат роботи програми Process Monitor

Щодо аналізу ключів реєстру програма Process Monitor надає таку унікальну можливість, як надання сумарної динамічної інформації за певний проміжок

часу у вигляді таблиці.

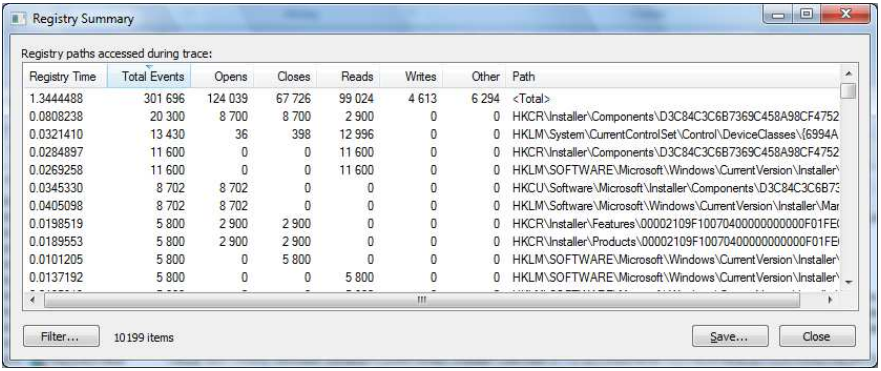


Рисунок 9.10 – Аналіз ключів реєстру

Якщо необхідно відстежити події доступу до тільки конкретної гілки, досить набудувати фільтр, вказавши, як шлях, відповідний ключ. Наприклад, для розділу реєстру HKEY_Classes_Root за певний проміжок часу буде виконано 8281 операцію, з яких:

- 144 – доводиться на відкриття ключа;
- 147 – доводиться на закриття ключа;
- 7988 – доводиться на операцію читання;
- 0 – доводиться на операцію запису параметрів (значень параметрів);
- 2 – на інші операції.

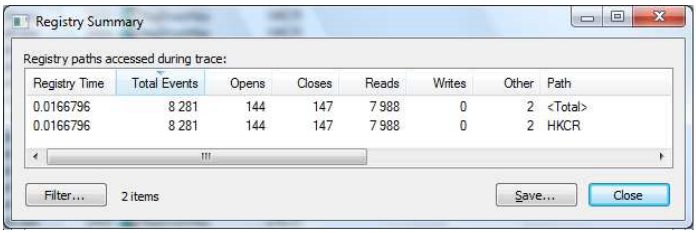


Рисунок 9.11 – Відстеження події доступу до конкретної гілки

Таким чином, установлюючи відповідне значення ключа у фільтр, можна акцентовано відстежувати операції тільки з даним ключем.

За кожною подією доступу до реєстру додатково можна отримати **загальну** інформацію, до якої належать:

– системний час, коли був здійснений доступ (Time of Day);

– ім’я процесу, який здійснив доступ до ключа реєстру (Process Name);

– ідентифікатор процесу (PID);

– найменування функції API, яка застосована для виконання дії (Operation);

– найменування ключа (Path).

Також можна отримати **детальну** інформацію у вікні властивостей (Ctrl+P):

- ідентифікатор потоку (Thread);
- результат операції (УСПІХ, ВІДМОВА, ІМ’Я НЕ ЗНАЙДЕНЕ і т.д.);
- тривалість операції (Duration).

RegShot. Утиліта призначена для фіксації змін у реєстрі Windows. Вона може робити знімок реєстру, зберігати його у файлі, завантажувати з файлу, порівнювати два знімки, знаходити всі відмінності (що змінилося, що було видалене, що з’явилося нового).

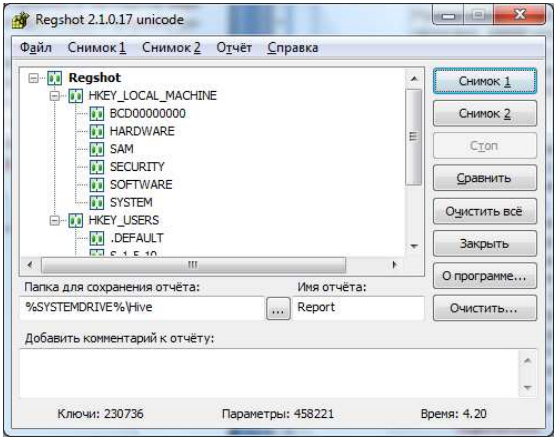


Рисунок 9.12 – Утиліта RegShot

За наслідками змін формуються декілька звітів. HTML звіт містить в 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 в стан, відповідний більш ранньому/пізньому знімку. Два звіти у форматі інсталяційних файлів undo.inf і redo.inf містять аналогічну інформацію, для аналогічної мети. Кількістю формованих звітів можна управляти за допомогою налаштувань. Утиліта не інтегрується в систему, не записує в реєстр інформацію, всі її налаштування зберігаються у файлі regshot.xml.

RegShot може стати в нагоді, якщо система почне нестабільно працювати, і необхідно з'ясувати, які зміни в реєстрі призвели до цієї нестабільності.

Той самий спосіб може бути застосований для продовження працездатності shareware програм, які працюють обмежений період часу, наприклад 30 днів, а після необхідна реєстрація.

Далі наведено приклад звіту, що формується програмою RegShot:

Звіт		
	Знімок A	Знімок B
Дата знімка	09.04.2014 18:40:36	09.04.2014 18:41:52
Комп'ютер	AAAA	AAAA
Користувач	BB	BB
SID користувача	S-1-5-21-2205154627-2141273298-1179179250-1000	S-1-5-21-2205154627-2141273298-1179179250-1000
Тип знімка	Реєстр повністю	Реєстр повністю
Час знімка	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
Файл відновлення реєстру .inf	–	–

Regcleaner – російська версія програми очищення реєстру Windows. За допомогою цієї програми можна підвищити продуктивність системи, звільнивши реєстр від залишків некоректно видаленого програмного забезпечення (рис. 9.13).

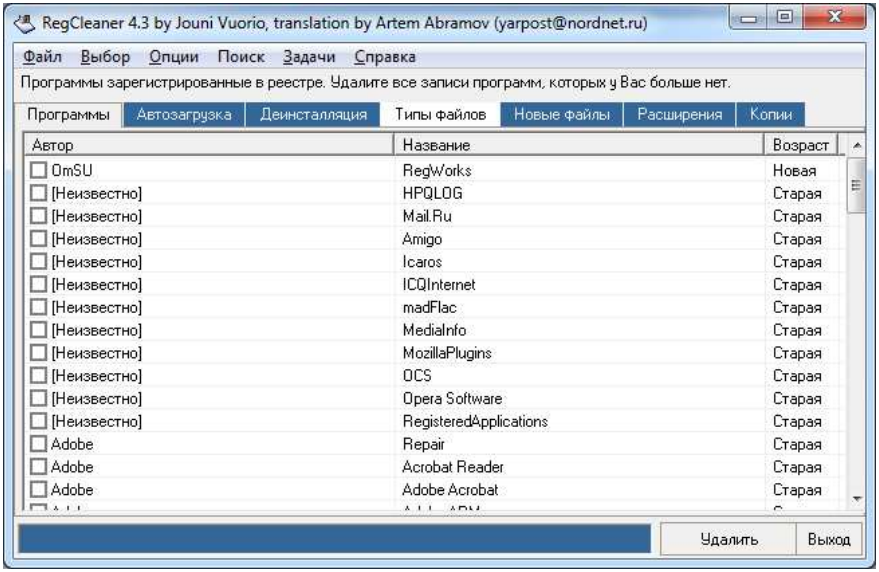


Рисунок 9.13 – Програма Regcleaner

RegCleaner зручний тим, що не дає можливості користувачеві внести зміни, що призводять до втрати працездатності системи або появи помилок, що рідко вдається уникнути при роботі з незрозумілою для більшості користувачів структурою запису даних у реєстрі.

Програма, яка дозволяє видаляти застарілі записи в реєстрі Windows (наприклад, при видаленні програмного забезпечення, можна знайти залишки програми). У інтерфейсі **RegCleaner** знаходяться сім вкладок: програми, автозавантаження, деінсталяція, типи файлів, нові файли, розширення, а також резервне копіювання. У кожній колонці можна побачити автора і назву програми, шлях до файлу, джерело, розширення і опис команди, а також дату і час резервних копій.

Переваги:

– ретельний аналіз реєстру;

– структура інтерфейсу виконана в закладному стилі, що вельми зручно;

– можливість вибору ручного або автоматичного чищення реєстру;

– зручна система виділення;

– інформативні підказки у верхній частині програми;

– можливість створення резервної копії реєстру перед чищенням;

– складний і ефективний алгоритм, який програма використовує для пошуку і очищення застарілих записів;

– сканування реєстру на наявність невживаних DLL-файлів;

– підтримка російської мови в інтерфейсі, що спрощує роботу;

– нескладні налаштування в інтерфейсі;

– розділ управління автозавантаженням;

– після чищення реєстру значно зменшується час завантаження операційної системи і помітно підвищується продуктивність.

Недоліки:

– в ручному режимі очищення не 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;

– виявлені проблеми в роботі програми на багатоядерних процесорах.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

ІНДИВІДУАЛЬНІ ЗАВДАННЯ

ВАРІАНТ 1

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\PnP
2. Розширення: .bmp; .js; .evt
3. Найменування 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: NoTrayContextMenu).

ВАРІАНТ 2

1. HKEY_CURRENT_USER\Control Panel\Mouse

2. Розширення: .m1v; .der; .png

3. Найменування розділу: HKEY_CLASSES_ROOT\Installer

4. Shareware додаток для інсталяції: adkm.exe

5. Приховання значків дисків у вікні Мій комп'ютер і Провідник і заборона на доступ до вмісту вибраних дисків (Гілка HKEY_CURRENT_-USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer, параметр NoDrives і NoViewOnDrive)

ВАРІАНТ 3

1. HKEY_USERS\.DEFAULT\Control Panel\Colors

2. Розширення: .aac; .mpe; .gadget

3. Найменування 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. Найменування розділу: HKEY_CURRENT_USER

4. Shareware додаток для інсталяції: MemopumpSetup.zip

5. Діалогове вікно відкриття і збереження файлу. (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. Найменування 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. Найменування розділу: HKEY_CURRENT_USER\Software
4. Shareware додаток для інсталяції: EXE-extractor.exe
5. Диспетчер завдань Windows XP і Синій Екран Смерті Windows XP (Гілка 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. Найменування розділу: HKEY_CURRENT_USER\Control Panel
4. Shareware додаток для інсталяції: mnkPlus.exe
5. Доступ до налаштувань мережі (див. табл. 9.7).

ВАРІАНТ 8

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa
2. Розширення: .gmmp; .au; .fdf

3. Найменування розділу: HKEY_LOCAL_MACHINE

4. Shareware додаток для інсталяції: MSItoEXECreatorDemo.exe

5. Управління можливостями "Робочого столу" (див. табл. 9.10).

ВАРІАНТ 9

1. HKLM\SYSTEM\CurrentControlSet\Enum\STORAGE

2. Розширення: .css; .pbk; .jpe

3. Найменування розділу: HKEY_LOCAL_MACHINE\Software

4. Shareware додаток для інсталяції: OE-Mail recovery.exe

5. Управління годинником, яке включає: синхронізацію системного годинника, вибір time-серверов, прикрасу годинничків. (Гілки: HKLM\SYSTEM\ControlSet001\Services\W32Time\TimeProviders\NtpClient, HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\DateTime\Servers, HKCU\Control Panel\International. Ключі: SpecialPollInterval, sTimeFormat).

ВАРІАНТ 10

1. HKEY_USERS\.DEFAULT\Identities

2. Розширення: .msu; .iso; .asx

3. Найменування розділу: HKEY_USERS

4. Shareware додаток для інсталяції: phones.exe

5. Дискові операції з перевірки диска: автоматичне виправлення помилок, зміна часу очікування. (Гілки: HKEY_USERS\.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. Найменування розділу: HKEY_CLASSES_ROOT

4. Shareware додаток для інсталяції: PrvDisk.exe

5. Повідомлення при завантаженні, автозавантаженні. (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. Найменування розділу: HKEY_CLASSES_ROOT\Installer

4. Shareware додаток для інсталяції: RecoveryToolboxForAccessInstall.exe

5. Заборона на доступ до вмісту вибраних дисків (гілка HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer. Ключ NoViewOnDrive). Організувати діалог з вибору найменувань маскованих дисків.

ВАРІАНТ 13

1. HKCU\Software\Microsoft\Windows\CurrentVersion\GroupPolicy\GroupMembership

2. Розширення: .cpl; .ifo; .nfo

3. Найменування розділу: HKEY_CLASSES_ROOT\Setting

4. Shareware додаток для інсталяції: setup_akl.exe

5. Дискові операції, такі, як зміна порогу видачі попередження про недолік вільного місця на диску, дефрагментація (гілка 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\Session Manager\Memory Management, HKLM\ SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management.

Ключі: DiskSpaceThreshold, NoLowDiskSpaceChecks, ContigFileAllocSize, DisableScreenSaver, Enable,ClearPageFileAtShutdown, DisablePagingExecutive).

ВАРІАНТ 14

1. HKLM \SYSTEM\CurrentControlSet\Enum\DISPLAY

2. Розширення: .reg; .adt; .dwfx

3. Найменування розділу: HKEY_CURRENT_USER

4. Shareware додаток для інсталяції: RecoveryToolboxForOutlookExpressInstall.exe

5. Видалення стрілки з ярличка, "долоньки" з відкритими ресурсами, збільшення швидкості спливання меню. (Гілка HKCR\lnkfile, HKEY_CLASSES_ROOT\piffile, HKCR\Network\SharingHandler, HKEY_CURRENT_USER\ControlPanel\Desktop. Параметри IsShortcut, за умовчанням, MenuShowDelay).

ВАРІАНТ 15

1. HKLM \SYSTEM\CurrentControlSet\Control\Print\ Printers

2. Розширення: .hta; .pfx; .dqy

3. Найменування розділу: HKEY_CURRENT_USER\System

4. Shareware додаток для інсталяції: ScreenshotMakerPro_Setup.exe

5. Заборону запуску програм, редактора реєстру (Гілка HKEY_CURRENT_USER\SOFTWARE\Microsoft\ Windows\CurrentVerson\Policies\ Explorer і 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. Найменування розділу: HKEY_CURRENT_USER\Software

4. Shareware додаток для інсталяції: sim-card.exe

5. Доступу до налаштування екрану (див. табл. 9.5).

ВАРІАНТ 17

1. HKEY_LOCAL_MACHINE\HARDWARE\ACPI

2. Розширення: .cat; .htm; .msc

3. Найменування 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\Control Panel\Desktop. Ключі CoolSwitch, CoolSwitchRows і CoolSwitchColumns)

ВАРІАНТ 18

1. HKEY_USERS\.DEFAULT\Control Panel\Mouse

2. Розширення: .log; .flv; .cer

3. Найменування розділу: HKEY_LOCAL_MACHINE

4. Shareware додаток для інсталяції: swf-to-gif.exe

5. Доступ до налаштувань системи (див. табл. 9.6).

ВАРІАНТ 19

1.HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print\Monitors

2. Розширення: .dic; .inf; .msp

3. Найменування розділу: HKEY_LOCAL_MACHINE\Software

4. Shareware додаток для інсталяції: vtrain-en50.exe

5. Доступ до налаштувань принтерів (див. табл. 9.9).

ВАРІАНТ 20

1. HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer

2. Розширення: .jpg; .chm; .jtp

3. Найменування розділу: HKEY_USERS

4. Shareware додаток для інсталяції: wma-converter.exe

5. Доступ до налаштувань паролів (див. табл. 9.8).

ВАРІАНТ 21

1. HKEY_USERS\.DEFAULT\Control Panel\Desktop

2. Розширення: .ape; .icc; .url

3. Найменування розділу: HKEY_CLASSES_ROOT

4. Shareware додаток для інсталяції: wzipse40.exe

5. Екран вітання Windows (Гілка HKU\.DEFAULT\Control Panel\Desktop, значення 2 для ключів FontSmoothing, FontSmoothingType, і значення 1 для ключа FontSmoothingOrientation).

ВАРІАНТ 22

1. HKEY_CURRENT_USER\Environment

2. Розширення: .tif; .ini; .divx

3. Найменування 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 більший); $L – < (знак менший); $N – поточний диск; $P – поточний диск і шлях; $Q – = (знак дорівнює); $T – поточний час; $V – версія Windows; $$ – $ (знак долара).

ВАРІАНТ 23

1. HKLM\HARDWARE\RESOURCEMAP\PnPManager\PnpManager

2. Розширення: .rle; .dot; .jnt

3. Найменування розділу: HKEY_CLASSES_ROOT\Setting

4. Shareware додаток для інсталяції: lb404setup.exe

5. Паролі і безпека. (Гілка HKLM\SOFTWARE\Microsoft\Windows\ CurrentVersion\Policies\Network, параметри NoDialIn, DisablePwdCaching, HideSharePwds, NoFileSharing, NoFileSharingControl, NoPrintSharing, NoPrintSharingControl)

ВАРІАНТ 24

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB

2. Розширення: .sct; .diagcab; .img

3. Найменування розділу: HKEY_CURRENT_USER

4. Shareware додаток для інсталяції: memory-card-data-recovery-demo.exe

5. Реєстраційні дані (гілка HKEY_LOCAL_MACHINE\SOFTWARE\-Microsoft\Windows NT\CurrentVersion).

ВАРІАНТ 25

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\PnP

2. Розширення: .crd; .svg; .odc

3. Найменування розділу: HKEY_CURRENT_USER\System

4. Shareware додаток для інсталяції: mnkPlus.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: NoTrayContextMenu).

ВАРІАНТ 26

1. HKEY_CURRENT_USER\Control Panel\Mouse

2. Розширення: .msi; .pdf; .crl

3. Найменування 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 HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer, параметр NoDrives і NoViewOnDrive).

ВАРІАНТ 27

1. HKEY_USERS\.DEFAULT\Control Panel\Colors

2. Розширення: .avi; .ico; .ppt

3. Найменування 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. Найменування розділу: HKEY_LOCAL_MACHINE

4. Shareware додаток для інсталяції: OE-Mail recovery.exe

5. Діалогове вікно відкриття і збереження файлу. (Гілка HKCU\SOFTWARE\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. Найменування розділу: HKEY_LOCAL_MACHINE\Software

4. Shareware додаток для інсталяції: phones.exe

5. Обмеження режиму MS-DOS (див. табл. 9.10).

ВАРІАНТ 30

1. HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls]

2. Розширення: .jse; .blg; .group

3. Найменування розділу: HKEY_USERS

4. Shareware додаток для інсталяції: PrvDisk.exe

5. Диспетчер завдань Windows XP і Синій Екран Смерті Windows XP (Гілка HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System і HKLM\SYSTEM\CurrentControlSet\Servises\i8042prt\Parameters відповідно. Ключі DisableTaskMgr і CrashOnCtrlScroll).

Увага!

1. Перед початком редагування реєстру обов'язково підготуйте копію відповідної гілки реєстру.

2. Ніколи не видаляйте підрозділи і пари параметр-значення, які створені не Вами!

Етап 1. У даній частині кожен студент відповідно до свого індивідуального варіанта (**п.1**) повинен досліджувати певну гілку системного реєстру. В результаті дослідження заповнити таблицю 9.14.

Таблиця 9.14 – Дослідження гілки системного реєстру

№ п/п	Ключ (підключи)	Параметр	Значення	Пояснення
1				
2				
.				

Етап 2. Досліджувати права доступу до ключів розділу реєстру відповідно до свого індивідуального варіанта (**п.1**) за допомогою програми AccessEnum.exe. Результат подати у вигляді таблиці 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. Досліджувати розділ HKEY_Classes_Root у Windows 7 на предмет асоційованих застосувань прописаним у системі стандартним розширенням імен файлів відповідно до свого індивідуального варіанта (**п.2**). Результат записати до таблиці 9.16.

Таблиця 9.16 – Дослідження асоційованих застосувань в HKEY_Classes_Root

Розширення файлу	Значення параметра за умовчанням	Шлях до зіставленого застосування з параметрами
.txt	txtfile	%SystemRoot%\system32\NOTEPAD.EXE %1

Етап 4. Досліджувати задані для атозапуска при старті системи програми, відомості про які зберігаються в системному реєстрі. Для цього використовувати програму ***AutoRuns*** для Windows. Обов'язково розписати найменування гілок реєстру, в яких прописуються автоматичній завантажувані файли. Результати занести до таблиці 9.17.

Таблиця 9.17 – Монітор автозавантаження

Найменування	Опис	Виготівник	Шлях до файлу	Дата створення файлу

Етап 5. Досліджувати доступ до ключа реєстру, який вибирається довільно з розділу, вказаного в **п.3** індивідуального завдання, в реальному масштабі часу за допомогою програми ***Process Monitor***. Результати занести до таблиці 9.18.

Таблиця 9.18 – Дослідження ключа реєстру в реальному масштабі часу

Назва ключа	Ім'я процесу	PID	TID	Операція (функція)	Результат	Тривалість

Подати скриншот узагальненої статистики доступу до даного ключа впродовж 5 хвилин роботи у вигляді таблиці 9.19.

Таблиця 9.19 – Дослідження узагальненої статистики доступу до ключа реєстру в реальному масштабі часу

Назва ключа	Всього подій	Відкриття ключа	Закриття ключа	Читання	Запис	Інша операція

Етап 6. Розробити **2 reg-файли системного реєстру:**

1-й Reg-файл – для переміщення в реєстрі гілки, яка вказана 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 використовувати зображення прізвища студента в ASCII-кодах.

2-й UnReg-файл – для видалення внесених першим файлом ключів, параметрів і їх значень.

Етап 7. Досліджувати зміну налаштування реєстру при інсталяції shareware програмного забезпечення, заданого в **п.4** індивідуального завдання, використовуючи програму ***RegShot***. До інсталяції зняти з реєстру знімок 1. Провести інста-

ляцію. Зняти знімок 2. Провести їх порівняння. Відстежити гілки, ключі і параметри, які були додані (змінені) програмою, що інсталювалася.

У звіт помістити відредагований звіт, який генерується програмою ***RegShot.*** Під редагуванням розуміють відображення тільки розділів, які містять НЕНУЛЬОВІ значення.

Додаткове завдання

Розробити додаток, що дозволяє управляти вказаними в **п.5** елементами Windows. Управління повинне дозволити користувачеві як встановлювати, так і відміняти встановлення параметрів.

Контрольні запитання

1. З яких файлів складається реєстр? Де вони розташовані?
2. Структура реєстру.
3. Структура основного розділу HKEY_CLAUSES_ROOT.
4. Основні розділи і їх призначення.
5. Параметри ключів. Типи параметрів і їх значення.
6. Способи відновлення реєстру.
7. У чому специфіка гілок реєстру?
8. З якою метою використовується гілка HKEY_CURRENT_USER?

Лабораторна робота 10

ДОСЛІДЖЕННЯ СИСТЕМНИХ СЛУЖБ І ДРАЙВЕРІВ

Мета роботи: ознайомитись з одними з найбільш важливих структурних елементів Windows – системними службами і драйверами. Набути практичних навичок зі створення служб, їх інсталяції і реєстрації в системному реєстрі операційної системи, а також досконально вивчити оточення, в якому виконуються системні служби і драйвери.

Указівки з підготовки до виконання лабораторної роботи

Необхідно вивчити порядок роботи диспетчера управління службами і залежності між різними sys-файлами у Windows. Додаткову інформацію з підготовки до роботи можна отримати в джерелах:

1. Побегайло А. П. Системное программирование в Windows. – СПб.: БХВ-Петербург, 2006. – С. 605–658.

2. Солдатов В.П. Программирование драйверов Windows. – М.: Бином, 2004. (Глава 12).

Теоретичні відомості

Системна служба Windows (англійською – system service; надалі – просто служба) – це фоновий процес, який може запускатись і працювати без участі інтерактивного користувача.

Архітектура системних служб Windows передбачає три види компонентів (рис. 10.1). Ядром є *менеджер системних служб* – **Service Control Manager (SCM)**. API для «спілкування» з SCM експортує стандартна бібліотека Windows advapi32.dll. Підсистема служб завантажується на самому початку старту ОС з виконуваного файлу services.exe, однойменний процес можна спостерігати в списку активних процесів увесь час роботи комп’ютера. Бібліотека advapi32.dll спілкується з services.exe за допомогою RPC (Remote Procedure Call).

SCM запускається під час завантаження системи і працює, поки комп’ютер

не буде вимкнений. Менеджер системних служб взаємодіє, з одного боку, з *програмами, які управляють*, а з іншого – з системними службами.

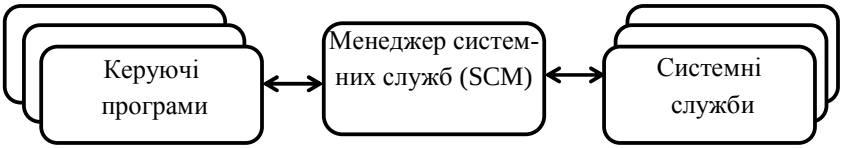


Рисунок 10.1 – Архітектура системних служб Windows

Відмінність служби від «звичайної програми» в тому, що служба, будучи запущеною, працює і після виходу користувача з системи навіть тоді, коли виводиться запрошення «Для входу до системи натисніть Ctrl+Alt+Delete». Тому в служби найчастіше оформляють такі програми, які повинні надавати послуги навіть тоді, коли немає користувачів, що увійшли до системи. До вбудованих служб Windows можна віднести:

- Spooler (служба друку);
- Alerter (служба сповіщення, часто використовувана системними адміністраторами);
- Server (дозволяє відкривати файли, named pipes та інші, комп'ютери іншими машинами в мережі) і т.д.

Основними завданнями менеджера SCM є:

- запуск при завантаженні системи тих служб, які запущені автоматично;
- зберігання конфігураційної бази даних з інформацією про всі служби;
- прийом запитів від управляючих програм і передача системним службам.

Windows визначає два види системних служб. Перший вид – це **служби режиму ядра** (kernel-mode services), які є ні що інше, як драйвери пристроїв.

Інший вид – служби Win32 – це звичайні Win32-процеси, що використовують спеціальний набір функцій для взаємодії з менеджером системних служб.

Керуючі програми – це звичайні Win32-додатки, які маніпулюють службами з використанням програмного інтерфейсу, що надається менеджером системних служб. Типовий приклад – це mmc snap-in «Служби» в Windows.

Драйвери пристроїв є завантажуваними модулями режиму ядра (як правило, це файли з розширенням .sys); вони утворюють інтерфейс між диспетчером введення-виведення і відповідним обладнанням. Ці драйвери виконуються в режимі ядра в одному з 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, як правило, пишуться на C (іноді на C++). Тому при правильному використанні процедур HAL вони є переносними між підтримуваною архітектурою Windows на рівні початкового коду, а на рівні двійкових файлів – усередині сімейства з однаковою архітектурою.

Існує 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 шин, інтерфейсів, пристроїв масової пам’яті і так далі;

− **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 віддзеркалення і шифрування дисків, перехоплення введення-виведення і деяку додаткову обробку інформації перед передачею її на наступний рівень;

− **мережеві редиректори і сервери**, що є драйверами файлових систем, які

передають запити файлової системи на введення-виведення іншим комп'ютерам у мережі і приймають від них аналогічні запити;

− **драйвери протоколів**, що реалізовують мережеві протоколи ніби TCP/IP, NETBEUI і IPX/SPX.

− **драйвери потокових фільтрів ядра**, що діють за ланцюжком для обробки потокових даних, наприклад при записі і відтворенні аудіо- та відеоінформації.

Оскільки установлення драйвера пристрою – єдиний спосіб додавання до системи стороннього коду режиму ядра, деякі програмісти пишуть драйвери просто для того, щоб дістати доступ до внутрішніх функцій або структур даних операційної системи, недоступних до режиму користувача (але документованих і підтримуваних в DDK).

Зараз усі драйвери до всіх пристроїв для Windows написані за технологією INF. У 2000-му році Microsoft зробила великий крок у питанні установлення додатків і «придумала» MSI-архівом cabinet-формату, до якого пришиті установочні скрипти в бінарному форматі з GUI-діалогами.

Скрипт – текстовий файл, що містить секції, параметри і значення параметрів секцій, які описують дії, необхідні для виконання інтерпретатором скрипту.

У системі має бути встановлений ***MSI Installer***, що є сервісом з правами системи і що тим самим дозволяє встановлювати багато пакетів навіть з правами користувача. Основними особливостями MSI-інсталлера, успадкованими від технології Active Setup з INF, є можливість створювати точки повернення реєстру, які застосовуються при деінсталяції пакета автоматично. Покликано в спробі позбавитися проблеми постійного руйнування і засмічення системного реєстру – великої проблеми у Windows. Відмінною особливістю MSI є прагнення створювати в реєстрі десятки унікальних **GUID-ів** для кожного додатка і навантажувати їх купою параметрів. *Скрипти INF нічим не відрізняються від простих bat-файлів* за принципом – задана 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 Windows присутні за умовчанням два інтерпретатори скриптів INF: **SETUPAPI** і **ADVANCEDINF**. ***Інтерпретатор*** – програма, що розпізнає синтаксис скрипта і послідовно виконує команди, вказані в скрипті.

Обидва інтерпретатори подають два DLL-файли в системній директорії і деяка кількість ключів у реєстрі. Інтерпретатор SETUPAPI знаходиться в бібліотечному файлі **setupapi.dll**, інтерпретатор ADVANCEDINF – в бібліотечному файлі **advpack.dll**. Бібліотеки інтерпретаторів не є виконуваними файлами, тому потрібен зовнішній ініціатор запуску функції інтерпретації скрипта. Ним є системна утиліта **RunDLL32.exe**.

Формат запуску будь-якої бібліотеки за допомогою RunDLL32:

rundll32.exe libraryname,EntryPoint parameters ,

де *libraryname* – ім'я файлу бібліотеки (бібліотеки інтерпретатора), допустимо вказувати без розширення, якщо бібліотека зареєстрована в списку SharedDLLs в системному реєстрі; *EntryPoint* – чутливе до регістру ім'я точки входу в бібліотеку (ім'я функції, що викликається), вказується відразу після коми, без пропусків; *parameters* – параметри, які передаються функції.

Синтаксис командного рядка для запуску інтерпретаторів скриптів INF побудований на цих же принципах. Формат запуску обумовлений високим ступенем дозволеності дій в системі файлу простого текстового формату. Microsoft не звикла довіряти скриптам. Тому, простий запуск подвійним клацанням у системі за умовчанням відкриває файл скрипта блокнотом, що достатньо легко змінити.

SETUPAPI

Основний інтерпретатор виконує основну масу дій:

- запис і видалення ключів, параметрів і значень системного реєстру;
- додавання і видалення рядків, заміна значень INI-файлів;
- розпаковування файлів з CAB-архівів, копіювання і видалення файлів;
- перейменування, зміна атрибутів файлів і папок;
- установлення драйверів;
- створення системних сервісів і пристроїв (Windows NT-based);

– перевірка призначених для користувача повноважень.

Приклад запуску інтерпретатора SETUPAPI для виконання скрипта:

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 обробки скрипта; *C:\Script.inf* – третій параметр для функції, що викликається, повний шлях до файлу скрипта. Зверніть увагу, потрібний повний шлях, оскільки проста вказівка імені файлу передбачає розташування файлу скрипта в системній директорії Windows. Ця ж примітка рівною мірою відноситься і до інтерпретатора ADVANCEDINF.

ADVANCEDINF

Надбудовний над SETUPAPI інтерпретатор, що дозволяє виконувати додаткові функції. Стандартні функції передає на виконання інтерпретатору SETUPAPI. Функції, які підтримуються інтерпретатором ADVANCEDINF:

- попередній запис змінних ключів реєстру в бінарний файл (функція повернення);
- одноразове виконання дій під кожним користувачем (доустановлення) при інсталяції і деінсталяції під час входу в систему (Active Setup);
- запуск виконуваних файлів з параметрами в прихованому і нормальному режимах;
- виведення простих діалогових вікон;
- читання директорій призначення операцій з файлами з реєстру.

Виходячи з вищесказаного, буде розумним віддавати перевагу інтерпретатору ADVANCEDINF – так користуватимемося перевагами обох інтерпретаторів. Виняток становить установлення драйверів – з цим справляється тільки SETUPAPI.

Приклад запуску інтерпретатора ADVANCEDINF для виконання скрипта:

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 файлу скрипта; *DefaultInstall* – другий параметр, ім'я виконуваної секції в INF-скрипті (зверніть увагу, ім'я секції нечутливе до регістру на відміну від точки входу); *4* – прапорець реакції інтерпретатора при обробці команд скрипта (табл. 10.1).

Таблиця 10.1 – Прапорець реакції інтерпретатора

Числове значення	Опис
0 або 128	Не перезавантажувати комп'ютер
1 або 129	Обов'язково і без питань перезавантажити комп'ютер
2 або 130	Запитати у 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, запитає у користувача перезавантажувати чи ні

Для розпізнання текстового файлу як файлу з INF-структурою інтерпретатор шукає в непорожніх рядках секційний заголовок [Version]. Заголовок традиційно розташовується на початку файлу і містить в своїй секції декілька рядків, що визначають тип скрипта. Заголовки бувають таких типів:

1. Стандартні.

2. Драйверні.

3. Розширені.

Тип скрипта указує, яким саме інтерпретатором слід виконувати скрипт. Залежно від інтерпретатора можна виконувати різний набір дій.

Стандартний заголовок. Скрипт із стандартним заголовком призначений для виконання операцій загального призначення. Набір функцій, підтримуваних скриптом із стандартним заголовком, визначається інтерпретатором, якому був переданий на виконання цей скрипт. Секція стандартного заголовка скрипта:

[Version]

Signature="$CHICAGO$"

SetupClass=BASE

ClassGUID={00000000-0000-0000-0000-000000000000}

Указані параметри *SetupClass* і *ClassGUID* рідко використовуються разом, зазвичай достатньо будь-якого з них. З драйверними заголовками ситуація декілька інша. Обмеження для використання стандартного заголовка: не рекомендується використовувати при написанні скрипту установлення драйверів. При побудові списку драйверів, відповідних типу встановлюваного пристрою, інтерпретатор відбирає скрипти драйверів саме за заголовком і драйверні скрипти із стандартним заголовком ніколи не будуть включені 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=03/16/2005,6.00.9830.1

Визначенням драйверного заголовка є параметри *Class* і *ClassGuid*. Вони вказують на один і той же тип драйвера, *Class* указує ім'я типу, *ClassGuid* – його GIUD. Список відомих поточній операційній системі типів драйверів можна так знайти в системному реєстрі:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Class

Класи драйверних типів наведено у таблиці 10.2.

Таблиця 10.2 – Класи драйверних типів

GUID	Клас
{4D36E966-E325-11CE-BFC1-08002BE10318}	Computer
{4D36E968-E325-11CE-BFC1-08002BE10318}	Display
{4D36E96B-E325-11CE-BFC1-08002BE10318}	Keyboard
{4D36E96A-E325-11CE-BFC1-08002BE10318}	HDC

GUID	Клас
{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 рекомендується вказувати тільки *Class*, оскільки не всі системи можна задовільнити. При визначенні скрипта як драйверного в системі ініціюється «вакцина» – реакція на зміну системної частини реєстру. Залежно від поточних налаштувань може бути виведений запит на підтвердження установлення драйвера. Починаючи з MS Windows XP (NT 5.1) під час налаштування за умовчанням створюється точка відновлення системної частини реєстру і змінених у ході установлення системних файлів. Крім того, при вказівці драйверного заголовка скрипт буде позбавлений можливостей, підтримуваних в AdvancedINF.

Розширений заголовок скрипта вказує, яка версія інтерпретатора AdvancedINF необхідна для виконання скрипта. Якщо поточна версія інтерпретатора нижче вказаного в скрипті, обробка скрипта буде перервана і виведеться помилка, в нашому випадку – «Error message»:

```
[Version]
Signature="$CHICAGO$"
AdvancedINF=2.0, "Error message"
```

Якщо файл з розширеним заголовком, призначений для обробки інтерпретатором AdvancedINF, буде в командному рядку переданий на виконання для SETUPAPI, виконаний він буде як файл із стандартним заголовком.

У кожному типі заголовка незмінним рядком є параметр Signature. Його

значення вказує, для якої ОС призначений скрипт. Відомо два значення: **$CHICAGO$** і **$WINDOWS NT$**. Перше призначене для всіх операційних систем Windows; друге тільки для Windows NT 5.0 (2000), 5.1 (XP), 5.2 (2003). Виняток становлять драйверні скрипти – необхідно точно визначити тип системи, інакше драйвер (залежно від типу) не буде розпізнаний як відповідний.

Знання драйвера включає знання про файли, які використовуються. Дані знання вказують на потрібність драйверу (сервісу). Одним з рішень цього питання є програма Dependency Walker (перегляд залежностей) – depends.exe. Dependency Walker – це програма, що дозволяє дізнатися, які DLL потрібні для запуску виконуваного файлу (і побачити все дерево залежностей), а також дізнатися, які функції він з них імпортує, та подивитись таблиці імпорту і експорту у DLL.

Головне вікно зображено на рис. 10.2, де наведена інформація про структуру бібліотеки kernel32.dll ОС Windows XP, що експортує 954 функції, написана на мові С.

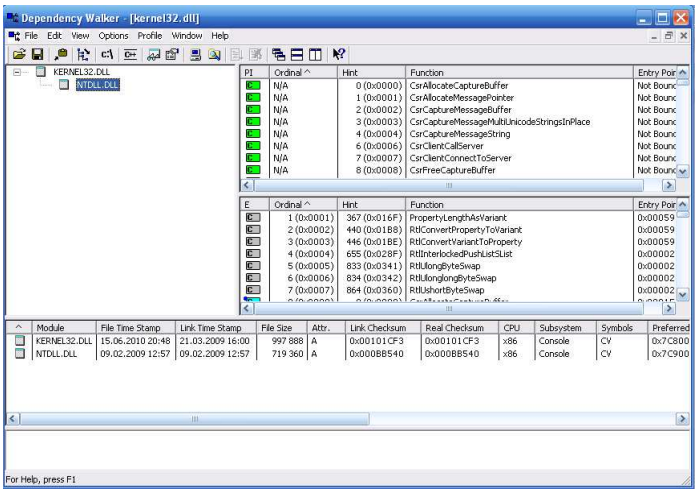


Рисунок 10.2 – Основне вікно утиліти depends.exe

Щоб побачити, які бібліотеки DLL, такі, як елементи управління ActiveX, знаходяться в процесі динамічного завантаження драйвера, скористайтеся **функцією профілізації** depends.exe. Потім перевірте додаток, щоб переконатися, що

були задіяні шляхи всіх кодів. Після завершення профілізації depends.exe відображає бібліотеки DLL, які були динамічно завантажені.

Використовуючи depends.exe, 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, які потрібні для підтримки додатка. Якщо під час запуску додатка на кінцевому комп'ютері виникають проблеми, можна скопіювати depends.exe на кінцевий комп'ютер і відкрити додаток в depends.exe. Depends.exe відображає бібліотеки DLL, яких бракує, а також бібліотеки несумісних версій.

Після отримання повного списку бібліотек DLL, від яких залежить додаток, можна визначити, які з цих бібліотек необхідно поширювати разом з додатком при його розгортанні на іншому комп'ютері. В більшості випадків немає необхідності в розповсюдженні системних бібліотек DLL.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Таблиця 10.3 – Індивідуальні завдання для виконання

Вар.	Inf-файл	Драйвер	Додаткове завдання
1	multiprt.inf	fltMgr.sys	Створити додаток з графічним інтерфейсом. Програма повинна запускати диспетчер для власного сервісу. Перед тілом функції оголошені всі глобальні імена, які надалі використовуються в сервісі. При цьому сервіс повинен відстежувати: – реєстрацію обробника керуючих команд для сервісу; – ініціалізацію структури стану сервісу

Продовження табл. 10.3

Вар.	Inf-файл	Драйвер	Додаткове завдання
2	net1394.inf	nwlnkflt.sys	Створення додатка, що встановлює зв'язок з менеджером сервісів і дозволяє відкривати доступ до бази даних сервісів. Тип доступу до бази даних повинен передбачати будь-яку комбінацію прапорців, у тому числі і установлення головних прав доступу
3	memstpci.inf	asyncmac.sys	Розробити графічний інтерфейс, що дозволяє виводити в діалоговому вікні властивості довільно вибраної користувачем служби із списку зареєстрованих на даному локальному комп'ютері. При цьому обов'язково забезпечити такі параметри: тип облікового запису, від імені якого запущено службу, що діє при виникненні першого, другого і подальшого перебою. За необхідності врахувати і надати опис вибраного сервісу на комп'ютері
4	rspndr.inf	wudfpf.sys	Створити додаток з графічним інтерфейсом, який дозволяє управляти правами доступу до довільно вибраного із зареєстрованих у системі системних служб. Забезпечити підтримку прапорців service_, read_, write_. Реалізувати можливість зміни прапорців в процесі роботи програми
5	dot4prt.inf	raspppoe.sys	Розробити графічний додаток, що дозволяє встановлювати призначені для користувача сервіси. Усередині сервісу забезпечити довільні імена, шляхи, окремий процес, особисту групу. Ці параметри повинні встановлюватися в НАЛАШТУВАННЯХ програми. Забезпечити можливість видалення ТІЛЬКИ встановлених, призначених для користувача сервісів
6	hidbth.inf	ipfltdrv.sys	Створити додаток з графічним інтерфейсом. Програма повинна запускати диспетчер для власного сервісу. Перед тілом функції оголошені всі глобальні імена, які надалі використовуються в сервісі. При цьому сервіс повинен відстежувати відкриття файлу протоколу роботи сервісу
7	dfrg.inf	mrxdav.sys	Розробити консольний додаток, що дозволяє управляти заздалегідь встановленою, призначеною для користувача службою. Під управлінням розуміють старт, стоп, паузу у виконанні,

Продовження табл. 10.3

Вар.	Inf-файл	Драйвер	Додаткове завдання
			відновлення, рестарт цієї служби. Супроводжувати вказані дії системою повідомлень і підказок
8	netepvcp.inf	ndistapi.sys	Розробити консольний додаток визначення стану сервісу за допомогою виклику функції QueryServiceStatus. Для цього необхідно зв'язатися з Service Control Manager з підключенням активної бази даних сервісів, відкриття сервісу, заздалегідь установленого в системі, визначення стану сервісу і розшифровка цього стану шляхом аналізу структури _SERVICE_STATUS
9	mfcem33.inf	tcpip6.sys	Створити додаток з графічним інтерфейсом. Програма повинна запускати диспетчер для власного сервісу. Перед тілом функції оголошені всі глобальні імена, які надалі використовуються в сервісі. При цьому сервіс повинен відстежувати ініціалізацію структури стану сервісу
10	ppa.inf	aec.sys	Створити додаток з консольним інтерфейсом, що дозволяє визначати конфігурацію сервера. Під конфігурацією сервісу розуміють інформацію, яка описує внутрішні і зовнішні імена сервісу, режими його запуску і роботи, що використовують функцію QueryServiceConfig. За висновками аналізу конфігурації вивести таку інформацію: тип сервісу, тип запуску, управління помилками сервісу, ім'я шляху, ім'я, що відображається
11	mdmrisa.inf	rasl2tp.sys	Створити додаток з графічним інтерфейсом. Програма повинна запускати диспетчер для власного сервісу. Перед тілом функції оголошені всі глобальні імена, які надалі використовуються в сервісі. При цьому сервіс повинен відстежувати установлення стану сервісу
12	digiasyn.inf	audstub.sys	Розробити графічний додаток, в якому продемонструвати визначення внутрішнього імені сервісу за його зовнішнім іменем за допомогою функції GetServiceKeyName. Під внутрішнім ім'ям сервісу розуміють те ім'я, під яким сервіс зберігається в базі даних сервісів, а також використовується менеджером сервісів для посилань на цей сервіс

Продовження табл. 10.3

Вар.	Inf-файл	Драйвер	Додаткове завдання
13	bthprint.inf	ipinip.sys	Розробити графічний інтерфейс, що дозволяє виводити в діалоговому вікні властивості довільно вибраної служби, призначеної для користувача зі списку зареєстрованих на даному локальному комп'ютері. При цьому обов'язково виводити такі параметри: внутрішнє ім'я служби, параметри, що відображаються, тип сервісу, тип запуску, прапорці контролю помилок, групу, до якої належить сервіс, шлях до файлу сервісу, а також його опис
14	bthspp.inf	psched.sys	Розробити графічне застосування, що дозволяє блокувати і розблоковувати процес бази даних сервісів, а також визначати: заблокована база даних сервісів чи ні. Продемонструвати факт блокування бази шляхом створення нового процесу, який повинен здійснювати спробу доступу. Ця спроба повинна завершуватись успішно в розблокованому стані і неуспішно – в протилежному випадку
15	ramdisk.inf	netbt.sys	Розробити графічний додаток визначення стану сервісу за допомогою виклику функції QueryServiceStatus. Для цього необхідно зв'язатись з Service Control Manager з підключенням активної бази даних сервісів, відкриття сервісу, заздалегідь установленого в системі, визначення стану сервісу і розшифровка цього стану шляхом аналізу структури _SERVICE_STATUS
16	msrio.inf	ptilink.sys	Створити додаток, що управляє прапорцями SC_ типів доступу до Service Control Manager. Передбачити всі можливі комбінації прапорців. Реалізувати приклад взаємодії програми, що управляє, з Service Control Manager
17	swflash.inf	nwlnkfwd.sys	Створити додаток з графічним інтерфейсом, що дозволяє змінювати конфігурацію сервера. Забезпечити управління режимами роботи програми за допомогою команд меню. Під конфігурацією сервісу розуміють інформацію, яка описує внутрішні і зовнішні імена сервісу, режими його запуску і роботи, використовуючи функції ChangeServiceConfig. За висновками аналізу

Продовження табл. 10.3

Вар.	Inf-файл	Драйвер	Додаткове завдання
			конфігурації вивести інформацію: тип сервісу, тип запуску, управління помилками сервісу, ім’я шляху, ім’я, що відображається
18	oem0.inf	ip6fw.sys	Розробити графічний додаток, що дозволяє управляти заздалегідь установленою, призначеною для користувача службою. Під управлінням розуміють старт, стоп, паузу у виконанні, відновлення, рестарт цієї служби. Супроводжувати вказані дії виведенням відповідних повідомлень у робочій зоні вікна
19	msnike.inf	ndiswan.sys	Створити додаток з графічним інтерфейсом, який дозволяє управляти правами доступу до довільно вибраного із зареєстрованих у системі системних служб (драйверів). Забезпечити 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
20	netlanep.inf	atmarpc.sys	Розробити графічний додаток, в якому продемонструвати визначення зовнішнього імені сервісу за його внутрішнім іменем за допомогою функції GetServiceDisplayName. Під зовнішнім ім’ям сервісу розуміють те ім’я, яке використовується для посилань на сервіс у вікні їх управління
21	ntapm.inf	raspti.sys	Розробити графічний інтерфейс, що дозволяє виводити в діалоговому вікні властивості довільно вибраного драйвера зі списку зареєстрованих на даному локальному комп’ютері. Вивести такі параметри: внутрішнє ім’я драйвера, параметри, що відображаються, тип драйвера, тип запуску, прапори контролю помилок, групу, до якої належить драйвер, а також його опис
22	ovcomp.inf	mrxsmb.sys	Створити додаток з графічним інтерфейсом. Програма повинна запускати диспетчер для власного сервісу. Перед тілом функції оголошені всі глобальні імена, які надалі використовуються в сервісі. При цьому сервіс повинен відстежувати:

Продовження табл. 10.3

Вар.	Inf-файл	Драйвер	Додаткове завдання
			– реєстрацію обробника керуючих команд для сервісу; – ініціалізацію структури стану сервісу
23	srchasst.inf	ipnat.sys	Створити додаток, що встановлює зв'язок з менеджером сервісів і дозволяє відкривати доступ до бази даних сервісів. Тип доступу до бази даних повинен передбачати будь-яку комбінацію прапорців, у тому числі і установлення прав доступу адміністратора
24	netauni.inf	afd.sys	Розробити графічний інтерфейс, що дозволяє виводити в діалоговому вікні властивості довільно вибраної користувачем служби зі списку зареєстрованих на даному локальному комп'ютері. При цьому обов'язково забезпечити такі параметри: тип облікового запису, від імені якого запущено службу, дії, що виконуються службою при виникненні першого, другого і подальшого перебою. Надати можливість користувачеві самостійно встановлювати час (у секундах), через який відбуватиметься обнуління лічильника відмов, а також рестарт сервісу і рестарт комп'ютера
25	mscpqpa1.inf	secdrv.sys	Створити додаток з графічним інтерфейсом, який дозволяє управляти правами доступу до довільно вибраного із зареєстрованих у системі системних служб. Забезпечити підтримку прапорців service_, read_, write_. Реалізувати можливість зміни прапорців у процесі роботи програми
26	usbprint.inf	rspndr.sys	Розробити графічний додаток, що дозволяє встановлювати призначені для користувача сервіси. Усередині сервісу забезпечити довільні імена, шляхи, окремий процес, особисту групу. Ці параметри повинні встановлюватися в НАЛАШТУВАННЯХ програми. Забезпечити можливість видалення ТІЛЬКИ встановлених, призначених для користувача сервісів.
27	adm_port.inf	rasacd.sys	Створити додаток з графічним інтерфейсом. Програма повинна запускати диспетчер для власного сервісу. Перед тілом функції оголошені всі глобальні імена, які надалі використовуються в

Закінчення табл. 10.3

Вар.	Inf-файл	Драйвер	Додаткове завдання
			сервісі. При цьому сервіс повинен відстежувати відкриття файлу протоколу роботи сервісу
28	epstw2k.inf	wudfrd.sys	Розробити консольне застосування, що дозволяє управляти заздалегідь встановленою, призначеною для користувача службою. Під управлінням розуміють старт, стоп, паузу у виконанні, відновлення, рестарт цієї служби. Супроводжувати вказані дії системою повідомлень і підказок
29	vgx.inf	ndisuio.sys	Розробити консольний додаток визначення стану сервісу за допомогою виклику функції QueryServiceStatus. Для цього необхідно зв'язатися з Service Control Manager з підключенням активної бази даних сервісів, відкриття сервісу, заздалегідь встановленого в системі, визначення стану сервісу і розшифровку цього стану шляхом аналізу структури _SERVICE_STATUS
30	netgpc.inf	netbios.sys	Створити додаток з графічним інтерфейсом. Програма повинна запускати диспетчер для власного сервісу. Перед тілом функції оголошені всі глобальні імена, які надалі використовуються в сервісі. При цьому сервіс повинен відстежувати ініціалізацію структури стану сервісу

1. Аналітична частина

Провести аналіз структури одного з inf-файлів, установленого в операційній системі в директорії %SYSTEMROOT%:\ WINDOWS\inf\ і заданого в **п.1** індивідуального завдання.

Аналіз повинен включати опис секцій даного inf-файлу. При цьому слід урахувати, що у разі відсутності вказаного в п.1 файлу у Вашій операційній системі слід вибрати його аналог.

2. Дослідницька частина

Етап 1. Вивчити 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 віднести такі параметри:

– внутрішнє ім’я драйвера;

– ім’я драйвера, що відображається;

– тип драйвера;

– тип запуску драйвера;

– контроль помилок;

– група драйверів (сервісів), до якої належить заданий драйвер;

– тег;

– шлях до файлу драйвера;

– розмір файлу драйвера;

– обліковий запис, від імені якого запускається драйвер;

– порядок відновлення системи після перебою драйвера.

Етап 2. Побудувати дерево залежностей вказаного в **п.2** драйвера від системних бібліотек, використовуючи утиліту depends.exe.

3. Програмна частина – додаткове завдання

Використовуючи рекомендовану літературу, надати в консольному або графічному вигляді (задається в **п.3** індивідуального варіанта) додаток. Передбачити відомості про автора, групу і факультет.

Контрольні запитання

1. Що таке служба?
2. Назвіть основні операції при роботі 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, їх запуск, зупинку.

Лабораторна робота 11

ДОСЛІДЖЕННЯ ЗАСОБІВ ЗАХИСТУ ДАНИХ

Мета роботи: набуття практичних навичок з організації захисту файлів в операційній системі, проведення криптоаналізу, аудиту парольного захисту різних об'єктів операційної системи.

Ознайомитися з реалізацією шифрування, генерацією ключів, створення і перевірки цифрових підписів і інших криптографічних завдань засобами інтерфейсу CryptoAPI, а також набуття практичних навичок з організації і зберігання відкритих, особистих або сесійних ключів.

Указівки з підготовки до виконання лабораторної роботи

Необхідно вивчити порядок шифрування і дешифрування даних засобами операційної системи, механізми аутентифікації і цифрового підпису.

Додаткову інформацію з підготовки до роботи можна отримати з літератури:

1. Домашев А.В., Программирование алгоритмов защиты информации: учебное пособие / А.В. Домашев, В.О. Попов, Д.И. Правиков, И.В. Прокофьев, А.Ю. Щербаков – М. : Нолидж, 2000. – С.160–268.

Теоретичні відомості

Шифрування – спосіб перетворення відкритої інформації в закриту й навпаки. Застосовується для зберігання важливої інформації в ненадійних джерелах або передачі її за незахищеними каналами зв'язку. Суть шифрування полягає в запобіганні перегляду початкового змісту повідомлення тими, у кого немає засобів його дешифрування. Зашифрувати можна не тільки текст, але і різні дані від файлів баз даних і текстових процесорів до файлів зображень.

Захисна підсистема комп'ютера потребує реалізації ряду загальних функцій, пов'язаних з логічним перетворенням змісту об'єктів (функції логічного захисту). До таких функцій належать:

− алгоритми контролю цілісності об'єктів комп'ютерної системи;

− алгоритми аутентифікації або авторизації суб'єктів (або користувачів, які управляють суб'єктами);

− алгоритми підтримки конфіденційності змісту об'єктів (наприклад, об'єкта вторинної аутентифікації користувачів).

Міжнародні і національні стандарти описують ряд добре вивчених функцій захисного характеру, зокрема:

− алгоритми хешування MD2 і MD5, ГОСТ Р 34.11–94;

− алгоритми генерації і перевірки електронного цифрового підпису DSS і ГОСТ Р 34.10–94.

Усі ці алгоритми мають різну специфікацію викликів (зокрема, різну довжину аргументів) і, природно, логічно не сумісні між собою.

При організації територіально розподілених комп'ютерних систем у різних локальних сегментах можуть використовуватись функціонально однакові, але семантично різні функції логічного захисту (наприклад, обчислення функцій контролю цілісності може проводитись 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 і для довільного суб'єкта, що входить в комп'ютерну систему (наприклад, використання суб'єкта обчислення електронного цифрового підпису для фіксації цілісності інформації, яка передається в зовнішню мережу). У зв'язку з цим питання взаємодії з функціями логічного захисту вивчатиметься без прив'язки 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 цілісності об'єктів).

При проектуванні і реалізації суб'єктів комп'ютерної системи підхід, що історично склався щодо використання загального ресурсу, пов'язаний з викорис-

танням роздільних суб'єктів, які виконують загальні для деякої підмножини зовнішніх по відношенню до даного суб'єктів функцій (наприклад, динамічно завантажені бібліотеки – типу DLL для ОС Windows). Логічно розповсюдити даний підхід на функції реалізації захисту.

Проблему проектування суб'єктів, що реалізовують функції логічного захисту, можна розглядати в декількох аспектах:

1. Проблему їх ***оптимальної реалізації*** в рамках деякого суб'єкта комп'ютерної системи, до якого звертається решта суб'єктів за виконанням відповідних функцій (в даному випадку мова йде про задання оптимізації параметрів «швидкодія–пам'ять» при реалізації захисних функцій).

2. Проблему ***мобільності*** суб'єктів, які використовують захисні функції при зміні внутрішнього наповнення, що реалізовує захисні функції суб'єкта (мають на увазі задання максимальної переносимості суб'єктів, що використовують захисні функції на інші алгоритми, наприклад інший алгоритм шифрування).

3. Проблему ***коректного використання*** суб'єкта, що реалізовує захисні функції, з боку модулів, які викликаються (проблема коректного використання декілька ширша, ніж просто коректність суб'єктів, оскільки передача інформації до асоційованих об'єктів-даних при виклику функцій логічного захисту має на увазі вплив на асоціативні об'єкти суб'єкта, що викликається).

У тексті використовується термін «**BLOB» – блок послідовних даних,** пов'язаний з експортом (виведенням у зовнішнє середовище) і імпортом (введенням із зовнішнього середовища) ключів. У даному випадку модуль реалізації захисних функцій, згідно з термінологією Microsoft, іменується ***криптопровайдером – Cryptographic Service Provider (CSP).***

Існують різні методи шифрування, які використовують спеціальні алгоритми.

З основних методів шифрування можна виділити «симетричне» та «асиметричне». У двох випадках, знаючи алгоритм шифрування, потрібно мати «ключ» (частину коду), а в останньому – два ключі («відкритий» і «закритий»).

У лабораторній роботі подані 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 присвячена лабораторна робота, 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, поданий у таблиці 11.1.

Таблиця 11.1 – Порівняльний аналіз алгоритмів шифрування

Назва алгоритму шифрування	Заборонений файл 1.0.2.6	VSEncryptor	Crypt4Free	Universal Shield	CryptoLab	MEO Encryption Software	AxCrypt	Encrypt Files
AES-128		√			√		√	
AES-192		√						
AES-256		√		√				√
RC2		√						√
RC4		√						√
RC5		√						

Продовження табл. 11.1

Назва алгоритму шифрування	Заборонений файл 1.0.2.6	VSEncryptor	Crypt4Free	Universal Shield	CryptoLab	MEO Encryption Software	AxCrypt	Encrypt Files
RC6		√						√
DES		√			√			
3-DES						√		
DESX			√					
Triple DES (168)		√		√				
Triple DES (192)								√
Blowfish		√	√	√	√	√		√
Twofish (128)		√						
Twofish (256)								√
Serpent (128)		√						
Serpent (256)				√				√
Affine Shift					√			
ARCFOUR					√			
Caesar					√			
Camelia		√						
Cast (128)				√				
Cast (256)		√						√
Cobra128				√				
Enigma					√			

Закінчення табл. 11.1

Назва алгоритму шифрування	Заборонений файл 1.0.2.6	VSEncryptor	Crypt4Free	Universal Shield	CryptoLab	MEO Encryption Software	AxCrypt	Encrypt Files
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 – програма для шифрування файлів (рис. 11.1). За допомогою її можна шифрувати окремі файли за алгоритмом IDEA (International Data Encryption Algorithm). Цей алгоритм досить стійкий і надійний, він широко використовується в Європі. Програма може додати свій пункт у меню провідника, щоб можна було зручніше запускати шифрування.

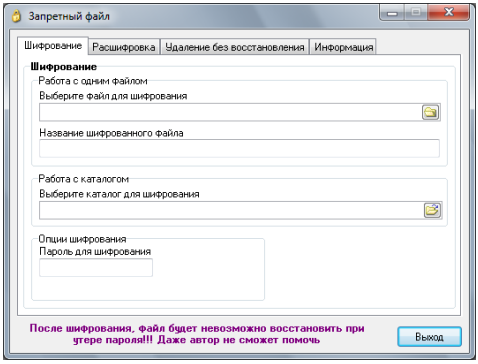


Рисунок 11.1 – Головне вікно програми Заборонений файл

VSEncryptor – це програма, призначена для шифрування файлу або тексту. Ви можете зашифрувати будь-який файл або текст за вашим вибором (рис. 11.2).

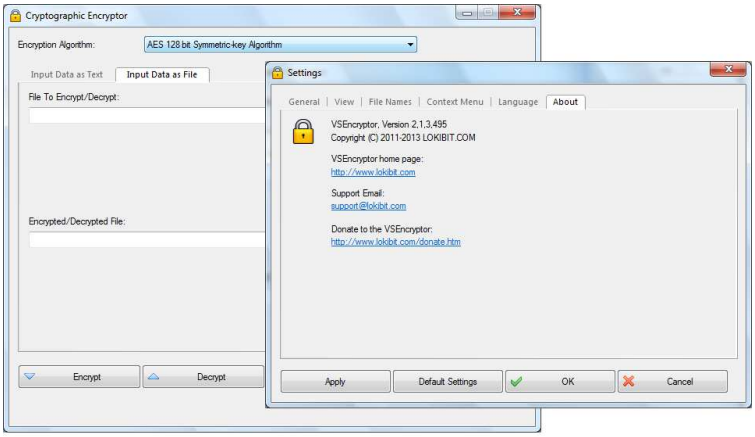


Рисунок 11.2 – Головне вікно програми VSEncryptor

VSEncryptor є серйозним рішенням, але в той же час проста у використанні і не вимагає яких-небудь технічних знань – просто виберіть (наприклад, перетягнувши файл в належне поле редактора) будь-який файл (або текст) і виберіть команду «Шифрувати».

Crypt4Free – дозволяє зашифровувати будь-які файли на будь-якому носієві, використовуючи один з двох наявних у програмі алгоритмів – Blowfish (448 бітовий ключ) і DESX (128 бітовий ключ) (рис. 11.3).

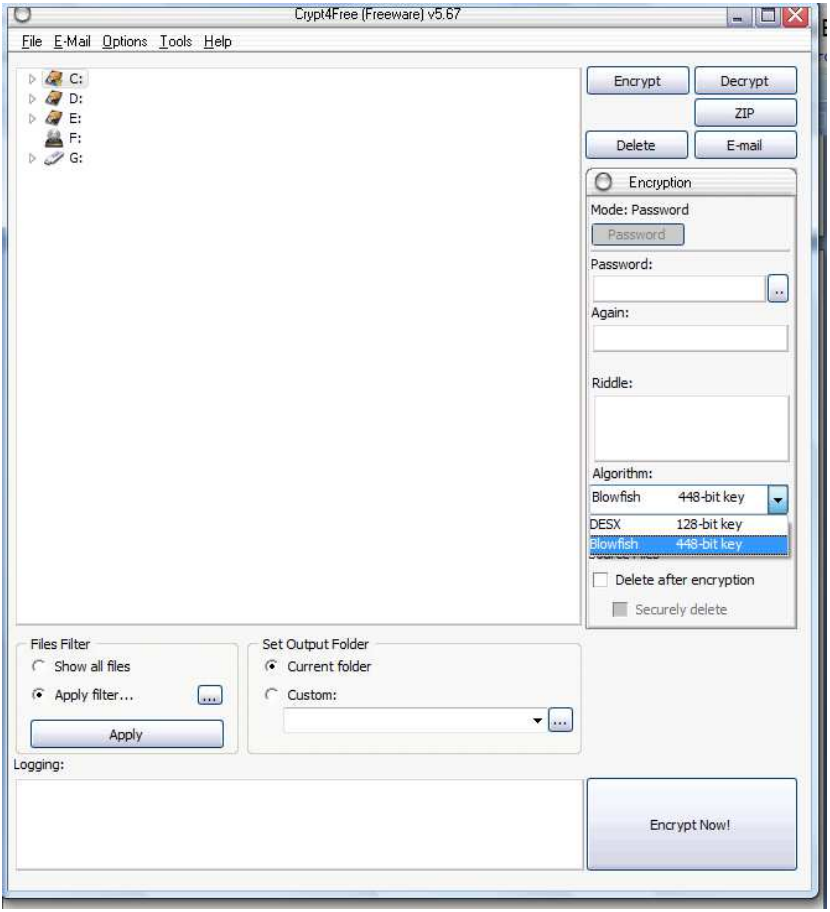


Рисунок 11.3 – Головне вікно програми Crypt4Free

Перед тим як зашифрувати файл, потрібно буде встановити пароль, його можна згенерувати автоматично або набрати самому на віртуальній клавіатурі (робиться це для захисту від кейлогерів), після чого натиснути кнопку «Start».

Підтримується одночасна з шифруванням упаковка файлів у zip-архіви; можливе швидке відправлення таких архівів по e-mail; можлива спільна робота з поштовою програмою для відправки текстових повідомлень в зашифрованому вигляді.

Universal Shield – це утиліта для захисту і шифрування файлів (рис. 11.4). Дозволяє зашифрувати, приховати файли, папки, диски і встановити додаткові права на доступ. Утиліта пропонує 9 алгоритмів шифрування, включаючи Blowfish, TRIPLE-DES, Rijndael та інші. Додаткові функції включають 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.

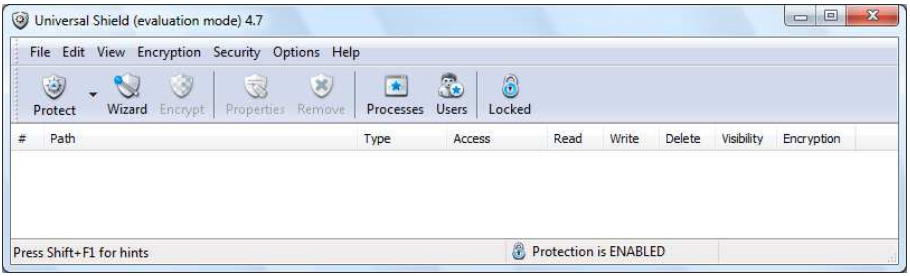


Рисунок 11.4 – Головне вікно програми Universal Shield

CryptoLab – це безкоштовна криптографічна сервісна програма для Microsoft Windows, яка взаємодіє з кодуванням, основаним на текстах/декодуванням з використанням плагін шифрувальних модулів (рис. 11.5). До покращених функцій належить SMTP підтримка електронною поштою, менеджер ключів і псевдовипадкова генерація ключів. Включені модулі кодування:

- Affine Shift.
- ARCFOUR.
- Blowfish шифр.
- Caesar шифр.
- Cost.

- Модулярний алгоритм перемикання (від EJC криптографії).
- Rijndael (американський стандарт кодування).
- Substitution шифр.
- Vigenre шифр.

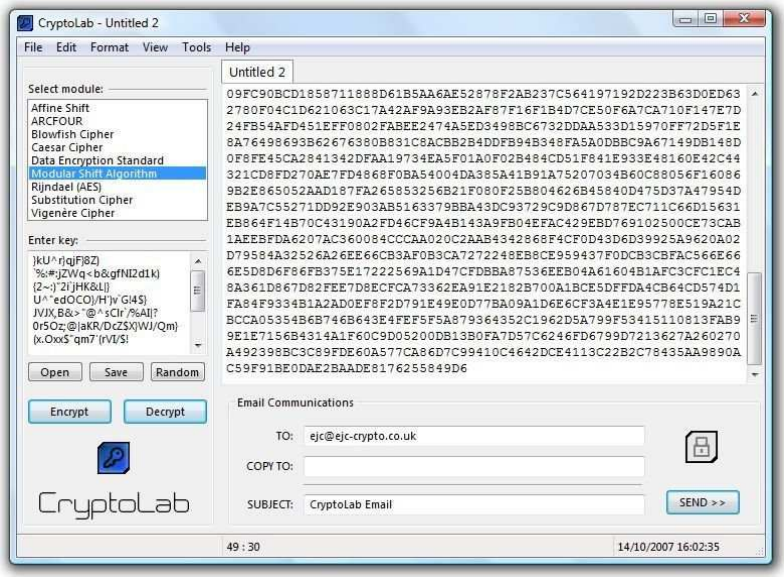


Рисунок 11.5 – Головне вікно програми CryptoLab

З CryptoLab можна кодувати текстові документи з військовими стандартами кодування – алгоритми, такі, як Rijndael – використовуються для кодування документів, класифікованих як документи держави США. Використовуючи CryptoLab, можна також установити високий рівень безпеки у вашій комп'ютерній системі.

Скачування і запуск даної програми в певних країнах нелегальний через певні криптографічні закони. EJC Cryptography не може надати легальну пораду.

Порада. Обов'язково потрібно проглянути відповідні закони країни за даною програмою. Так, наприклад, закони США на момент написання криптографічної програми не можуть бути експортовані до Афганістану, Куби, Ірану, Іраку,

Лівії, Північної Кореї, Судану, Сербії і Сірії.

MEO Encryption Software – є програмою для файлового та шифрування електронної пошти і дешифрування (рис. 11.6). Програма має основний графічний інтерфейс користувача тільки з трьома опціями: Зашифрувати файли, Зашифрувати електронну пошту або Дешифрувати файл.

Рисунок 11.6 – Головне вікно програми MEO Encryption Software

Перша опція дозволяє шифрувати файли для випробування на злом. Можна використовувати два алгоритми шифрування, щоб зашифрувати дані: стандартний шифр або 3-DES. 3-DES безпечніше, але займає більше часу. Можна створити файл самодешифрування, який запитує пароль перед вилученням. MEO не пропонує стискування для файлів.

Інша робота шифрування, яку можна виконати з MEO, відправлення зашифрованих листів, які не будуть прочитані, якщо неможливо дешифрувати їх. У додатку є сервер SMTP, який дозволяє йому відправляти зашифровані повідомлення зсередини додатка, і він навіть підтримує з'єднання.

Особливість MEO полягає в тому, що вона працює і в Windows, і в Mac OS X, таким чином, можна спільно використовувати файли з вказаними платформами.

AxCrypt – зручна і легка у використанні утиліта для шифрування будь-яких файлів (рис. 11.7). AxCrypt не додається в трей і не займає оперативну пам'ять.

Програма після установлення додається в контекстне меню файлів. Для захисту файлу від сторонніх очей досить клікнути правою кнопкою мишки, і в меню, що відкрилося, в розділі AxCrypt, вибрати пункт «Шифрувати». Буде запропоновано двічі ввести пароль (щоб не помилитися, інакше файл назад потім отримати не реально). Замість пункту «Шифрувати» можна вибрати «Шифрувати копію» або «Шифрувати копію .EXE». У першому випадку буде створена зашифрована копія файлу, що відкривається програмою AxCrypt (оригінал залишиться незайманим), в другому – зашифрована копія файлу, що відкривається на будь-якому комп'ютері без необхідності установлення програми (досить просто знати пароль).

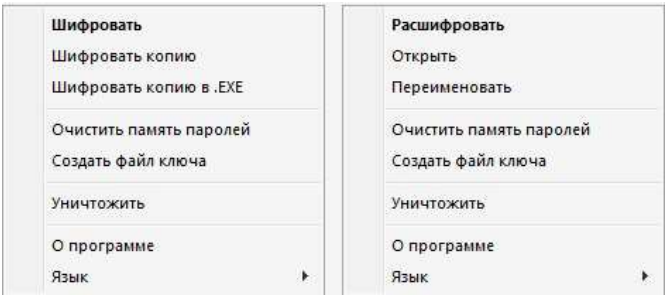


Рисунок 11.7 – Діалогове вікно програми AxCrypt

Алгоритми шифрування AES-128 і SHA-1. Після шифрування у файла з'являється новий значок і розширення. Програма AxCrypt інтегрується у Windows і з'являється в контекстному меню, яке відкривається кліком правої кнопки миші, потім потрібно вибрати AxCrypt і натиснути «Зашифрувати». Інтерфейс AxCrypt мінімальний, підтримує 12 мов, серед яких російська. Але враховуючи простоту, програмою користуватися зручно і без цього.

Програма не займає багато оперативної пам'яті, не вимагає особливих знань для користування. В опціях є такі можливості, як шифрування, розшифрування, редагування, стискування і перегляд файлів. AxCrypt має функцію «захист клю-

чем». Файл відкриється тільки тоді, коли в нього завантажать ключ, який може знаходитися завжди з користувачем, разом з USB носієм.

Encrypt Files – програма шифрування файлів у вільному доступі, щоб надійно зашифрувати і захищати конфіденційні дані (рис. 11.8). Швидке і легке програмне забезпечення, встановлене на комп'ютері, яке підтримує 13 просунутих алгоритмів шифрування, робить файли прихованими після шифрування. Encrypt Files дозволяє також регулювати доступ до зашифрованих файлів.

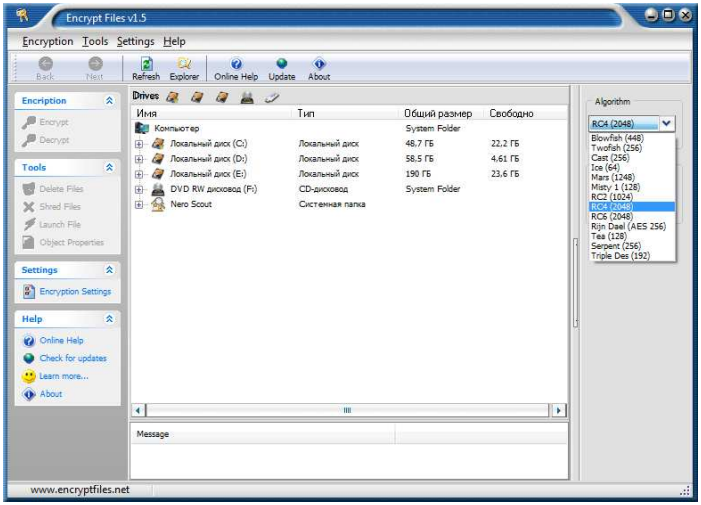


Рисунок 11.8 – Діалогове вікно програми Encrypt Files

Загальною особливістю всіх перерахованих програмних продуктів є використання паролів (key), тобто ключової послідовності, яка бере участь у побудові зашифрованої послідовності.

Пароль – 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 домена, доступ до яких користувач дістає, пред'явивши правильний логін і пароль. Надалі буде розглянуто інструменти, які може використовувати і хакер, що побажав діс-

татися до чужих таємниць, проте такі інструменти для нього в більшості випадків не самий кращий варіант. Для користувача, що забув пароль, завдання зводиться тільки до його отримання. Перед хакером, що цікавиться закритою інформацією, постає задача інша і ширша – зазвичай він шукає уразливість, знання якої дозволяє отримати контроль над чужим комп'ютером.

ПАРОЛІ ПРОДУКТІВ MICROSOFT OFFICE

Наступним важливим завданням може стати відновлення паролів до документів Microsoft Office. В цьому випадку необхідно скористатись програмним забезпеченням Advanced Office Password Recovery (рис. 11.9). Атака на пароль здійснюється за допомогою словника, за маскою, а також прямим перебором.

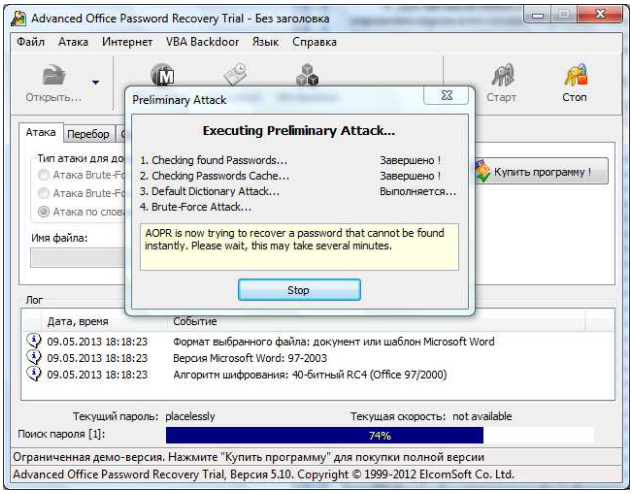


Рисунок 11.9 – Атака прямим перебором

Люди використовують деякі слова як паролі частіше за інших. Частотні словники перераховують найбільш популярні слова. Хороші словники містять сотні тисячі слів. Хороший частотний словник містить назви корпорацій, заголовків кінофільму, торгові марки і так далі. Злом за допомогою словника зазвичай відбувається дуже швидко, навіть якщо використовується величезний словник. У пакет PwlTool не включаються словники. Можна використовувати відносно малі слов-

ники (~80 КБ), словник від http://webdon.com/DownLoad/DIC1.ZIP або величезний (~9 МБ) http://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» включений, RePwl спробує всі варіанти подібно wordXXX, де слово – слово від словника, і XXX – суфікс, сформований згідно з параметрам налаштування рішення «в лоб». **Важливо! Переконайтесь, що всі слова в словнику знаходяться у верхньому регістрі.**

Атака «в лоб» (пошук усіх можливих паролів) не підходить для довгих паролів, тому що потрібно багато часу. Головним чином є комбінації подібні jkqmzwd, які є повністю безглуздими серед мільярдів і квінтильйонів обшукуваних паролів. Посилена «атака в лоб» – оптимізований алгоритм дослідження, який тільки пробує «розумні» паролі. В результаті час злому скорочується.

Але є 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 даного алгоритму;
- деякі слова важкі для такого алгоритму атаки. Ви повинні визначити рівень як ціле число в діапазоні 1 ... 26. Розумні значення для даного типу атаки – 9 ... 16 (значення за умовчанням – 13).

При спробі відновлення паролів на відкриття документа Office 2007 і Office 2010 варто врахувати, що в даному випадку для шифрування документів застосовується алгоритм AES з довжиною ключа 128 розрядів, а також хешування за алгоритмом SHA-1 (паролі до Word, Excel, PowerPoint, Access). Тільки прості і короткі паролі можуть бути виявлені простим перебором. Крім того, при відновленні паролів Office 2010 процес відновлення буде повільніший, ніж при відновленні паролів Office 2007. Для прискорення процесу відновлення паролів рекомендується застосовувати сучасні графічні карти від nVidia і AMD/ATI (рис. 11.10).

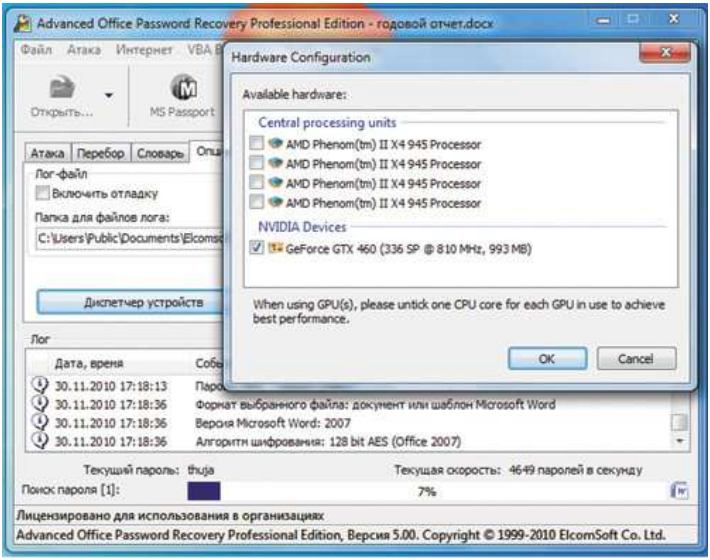


Рисунок 11.10 – Відновлення пароля до документа Office 2007 з використанням відеокарти

Для видалення пароля з документів Office призначено програмне забезпечення ***Advanced Office Password Breaker***.

Видалення парольного захисту можливе з файлів, які збережені у форматі .doc і .xls. Формати файлів Word 2007–2010 і Excel 2007–2010 не підтримуються. Головне вікно Advanced Office Password Breaker показане на рис. 11.11.

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 кількості ядер. Продукт Advanced Office Password Breaker Professional підтримує до 4 процесорів або ядер, дозволяючи прискорити процес перебору ключів в 4 рази. Корпоративна версія підтримує до 32 процесорів.

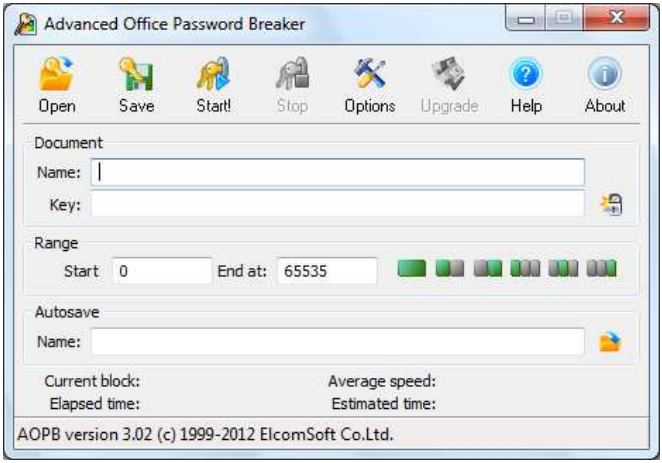


Рисунок 11.11 – Видалення пароля документа Word

Захист документів Microsoft Word і Microsoft Excel не завжди дозволяє швидко знайти забутий пароль. Використовуючи атаку за словником, прямий перебір паролів і інші методи, відновити пароль можна, проте цей процес може зайняти багато часу. Разом з тим, у випадку якщо захист документа сумісний з форматом Microsoft Office 97, існує рішення, що дозволяє гарантовано відкрити документ протягом невеликого проміжку часу. Даний формат захисту уразливий, оскільки використовується ключ шифрування довжиною всього 40 розрядів.

Для прискорення процесу перебору 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, використовуються класичні райдужні таблиці. Вірогідність розшифровки документа за декілька хвилин становить 97 %.

Програми підбору паролів до продуктів Microsoft Office можна скачати із спеціалізованих сайтів, наприклад:

- http://www.passwords.ru/,
- http://www.elcomsoft.com/,

− http://lastbit.com/mso/default.asp.

На сайті http://www.passwords.com/ розміщені посилання на ресурси, належать до теми паролів. Наприклад, дві програми з представленого списку:

- http://www.passwords.ru/aopb.html Advanced Office Password Breaker (AOPB) – програма розшифровки документів Word і Excel, захищених паролем на відкриття документа. Забезпечує гарантоване відкриття документа, незалежно від складності пароля і його довжини, що досягається шляхом перебору 40-бітових ключів шифрування. Така невелика довжина ключа використовується в Microsoft Office через експортні обмеження.
- AOPR – програма для відновлення втрачених паролів до документів Microsoft Office. Поставляється в двох редакціях – Standard і Professional. Більшість паролів знаходять миттєво, прямим декодуванням. Інтерфейс російський і англійський (рис. 11.12).

Паролі, що захищають архівні файли та файли з документами різних програмних продуктів, як правило, не зберігаються в документах в зашифрованому вигляді. Пароль використовується як ключ шифрування і ніде не зберігається.

При використанні утиліти Advanced Office Password Recovery можливе відновлення паролів усіх документів. Не дивлячись на те, що MS Word, шифруючи документ, використовує досить складний алгоритм (пароль не зберігається усередині файлу), він може бути розкритий з використанням методу «грубої сили» або атаки за словником. Середня швидкість при використанні процесора Intel становить 1 млн паролів у секунду.

В Excel використовуються аналогічні технології захисту, що і в Word. Відмінність становлять тільки паролі на книги/листи. Хеш-кодування паролів зберігається в документі. Найпарадоксальніше, що довжина хеш-кодування всього 16 біт. Отже, для кожного хеш-кодування існує 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.

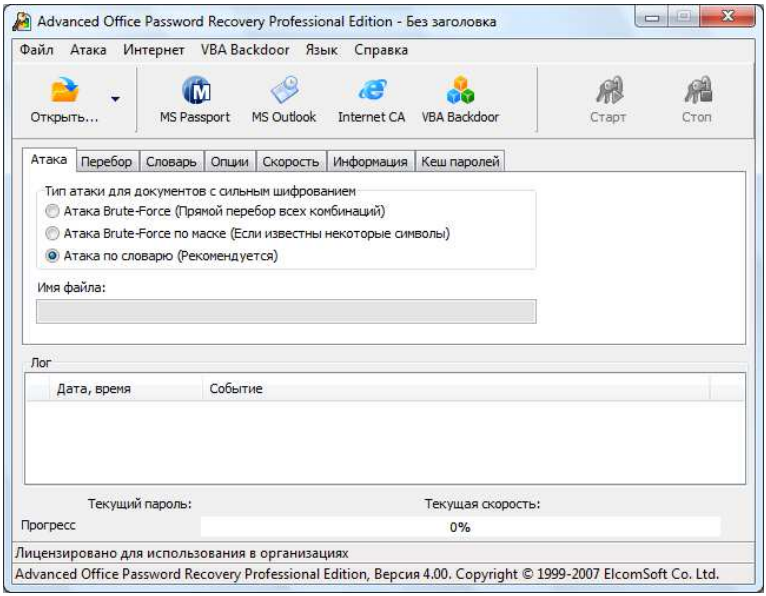


Рисунок 11.12 – Вікно програми Advanced Office Password Recovery

Паролі зберігаються в заголовку файлу. Заголовок зашифрований за RC4, але ключ шифрування має довжину 32 біт і зберігається в одній з системних DLL. Знаючи цей ключ, можна знайти будь-який пароль на базу Access.

ПАРОЛІ ПРОДУКТІВ ADOBE READER

Для злому використовується програмний продукт ***Advanced PDF Password Recovery*** (рис. 11.13). Дана версія підтримує роботу з документами, пароль на яких був установлений в Adobe Acrobat, починаючи з версії 3.x і закінчуючи 9.x, також є підтримка роботи з 128-bit RC4 шифруванням і 256-bit AES шифруванням, можливе зняття всіх наявних обмежень з документа формату PDF, можна вести розшифровку документа, якщо пароль відкритий і відомий, також можна вести атаку з перебору на відкритий пароль, Advanced PDF Password Recovery уміє атакувати на відкритий пароль, використовуючи для цього словники.

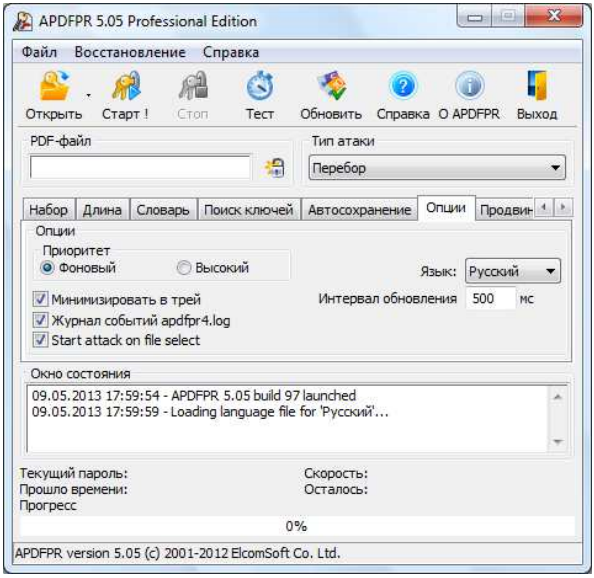


Рисунок 11.13 – Вікно програми Advanced PDF Password Recovery

Розробники гарантують відновлення пароля який використовував 40-бітове шифрування, і цей процес займе декілька днів. Якщо ви використовуєте багатоядерні процесори, то результат буде швидший, також можна прискорити процес, якщо вбудований GPU прискорення для відеокарт NVIDIA.

Advanced PDF Password Recovery гарантовано відновить пароль 40-розрядного шифрування з використанням технології Thunder Tables за декілька хвилин, дана функція доступна в максимальній версії за 12 т.р, яка і подана.

ПАРОЛІ АРХІВАТОРІВ

На сьогоднішній день існує досить багато алгоритмів криптографічного захисту інформації в архіваторах. Проте в переважній більшості архіваторів реалізований, як правило, який-небудь один метод. Мова йде про найбільш поширені архіватори. ZIP-кодування, як і шифрування за алгоритмом DES (Data Encryption Standard), складно назвати стійким. Найбільш надійним зараз є алгоритм AES, прийнятий як стандарт у США в 2001 році. Якщо подивитися на найпопулярніші

в країнах СНД архіватори, то це, без сумніву, WinZip і WINRAR. Розглянемо шифрування в цих архіваторах трохи докладніше.

У архіваторі WinZip реалізовано три алгоритми шифрування:

- Standard Zip 2.0 encryption – використовується за умовчанням;
- 128-it AES encryption – криптографічний алгоритм AES з довжиною ключа 128 розрядів;
- 256-it AES encryption – криптографічний алгоритм AES з довжиною ключа 256 розрядів (посилений алгоритм шифрування).

Користувачі, як правило, застосовують перший алгоритм (за умовчанням), адже він є найбільш слабким. Якщо ви не знаєте, за допомогою якого архіватора одержувач вашого архіву розпаковуватиме його, задіється метод за умовчанням.

Разом з тим необхідно підкреслити, що який би алгоритм шифрування ви не використали, стійкість вашого шифру насамперед залежатиме від стійкості ключа. Отже, при використанні шифрування необхідно застосовувати стійкі паролі!

До недоліків шифрування WinZip потрібно також віднести те, що WinZip не шифрує коментарі zip-файлів і такі властивості зашифрованих файлів, як найменування, дати і так далі Адже це вельми цінна інформація для аналітика. Далеко не всі користувачі перейменовують файли, що архівуються, а вже тим більше змінюють решту атрибутів (дата створення, зміни, розмір і т. д.).

У архіваторі WINRAR застосовується алгоритм шифрування AES з довжиною ключа 128 розрядів. Варто відзначити, що файли, зашифровані у WINRAR, відкриватимуться і у WinZip. Проте зворотне буде правильне лише для алгоритму шифрування Zip 2.0. Крім того, потрібно запам’ятати, що для надійного пакування даних в архів з шифруванням, необхідно подбати про надійне видалення файлів з носія, на якому вони знаходилися.

Advanced Archive Password Recovery – програмне забезпечення призначене для відновлення паролів до архівів Zip, RAR, ACE, ARJ (рис. 11.14).

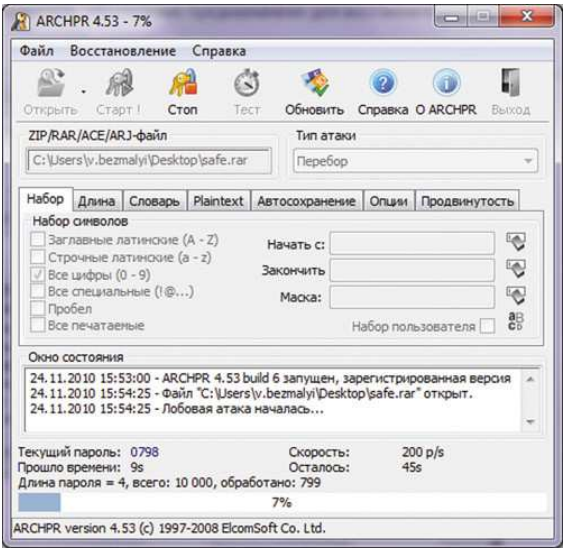


Рисунок 11.14 – Вікно програми Advanced Office Password Recovery

Відновлення пароля до архіву RAR забезпечується двома шляхами:

- знімає парольний захист;
- відновлює оригінальний текстовий пароль.

Підтримувані формати архівів – ZIP/PKZip/WinZip, RAR/WinRAR, ARJ/WinARJ, та ACE/WinACE (1.x), створені будь-якими архіваторами, й архіви, що саморозпаковуються, створені в PKZip, WinZip, RAR і WINRAR.

При використанні WinZip 8.0 і більш молодших версій гарантується зняття захисту протягом години. Крім того, варто відзначити, що за наявності хоч би одного файлу з архіву весь архів буде розшифрований за хвилину (для архівів у форматі ZIP і ARJ).

ВИКОНАННЯ КРЕКІНГУ ЗАХИСТУ ВИКОНУВАНОГО ФАЙЛУ ЗА ДОПОМОГОЮ OLLYDBG

Для виконання крекінгу виконуваного файлу буде потрібно:

1. Налагоджувач. Саме за допомогою цієї програми буде проводитися аналіз виконуваного коду тестової програми. Для прикладу виберемо **OllyDbg v.2.01.**

2. Нех-редактор. За допомогою цієї програми буде проводитися модифікація початкового коду. Для прикладу виберемо програму **Xvi32.**

3. Тестова програма, **TESTP.exe,** яку необхідне крекнути.

Усі вказані програмні забезпечення не вимагають інсталяції.

Спершу запускається налагоджувач 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** наведено на рис. 11.15.

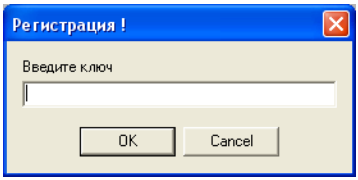


Рисунок 11.15 – Початкове вікно програми **TESTP.exe**

1. Відкриваємо програму **TestReg.exe** у налагоджувані OllyDbg. Для цього відкриваємо меню **File → Open** (F3) і вибираємо тестову програму. Після завантаження у вікні CPU буде надано лістинг, зображений на рис. 11.16.

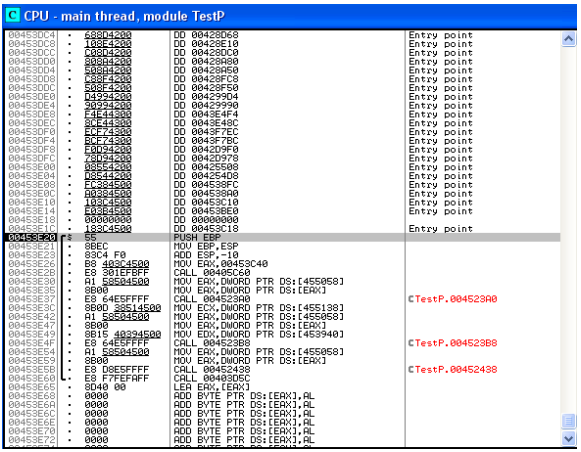


Рисунок 11.16 – Лістинг вікна CPU

Наведеним є асемблерний код тестової програми, наприклад:

PUSH EBP ; Початок іншої функції

CALL TestP.00405C60 ; Виклик функції

2. Тепер треба знайти функцію, яка видає вікно із запитом уведення ключа. Для цього виконуємо програму покроково натискаючи клавішу F8 доти, поки не з'явиться вікно із запитом введення. Після декількох натиснень з'являється діалогове вікно введення (рис. 11.17).

Відомо, що у функції (TestP.004523B8) виводиться діалогове вікно введення ключа. Необхідно знайти функцію, яка відображає це вікно. Перед рядком **CALL TestP.004523B8** треба поставити точку зупинки (Breakpoint). Для цього виділяємо рядок перед нею, натискаємо **F2** і бачимо, що у віконце Breakpoints додався рядок, наведений на рис. 11.18.

Далі проводиться рестарт програми **Debug → Restart** (Ctrl + F2) для того, щоб завантажити тестову програму наново. Після чого потрібно дійди до точки зупинки. Для цього натискаємо ▶ для того, щоб виконати нашу програму. В результаті покрокове виконання програми зупиниться на першій точці зупинки.

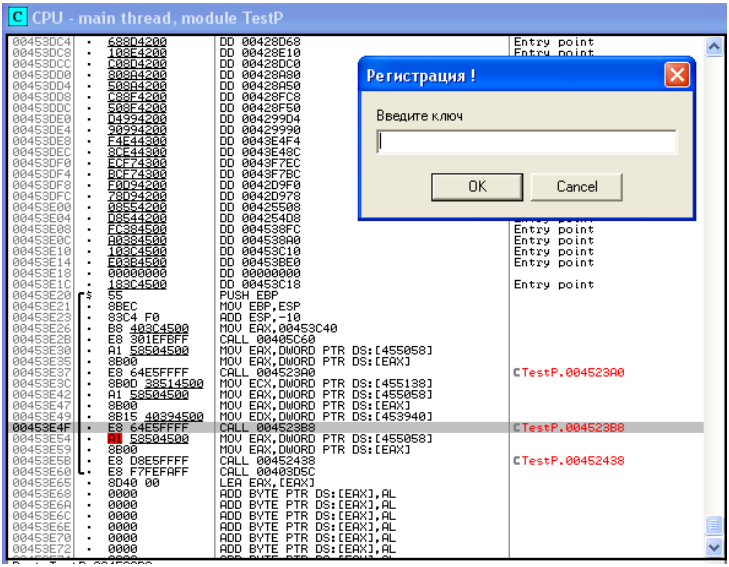


Рисунок 11.17 – Діалогове вікно вводу

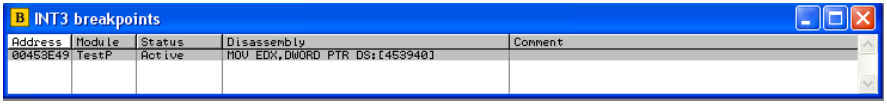


Рисунок 11.18 – Вікно Breakpoints

Для того щоб зайти у функцію натискаємо клавішу F7. Відмінність клавіш очевидна: F8 – трасує програму без заходу у функції, F7 – із заходом у функції. Результат показано на рис. 11.19.

Треба продовжувати, поки не знайдено функцію, яка виводить вікно, обробляє натиснення на кнопку OK і визначає, чи правильно введений ключ чи ні. Продовжувати натискати F8, поки не з’явиться вікно, зображене на рис. 11.20.

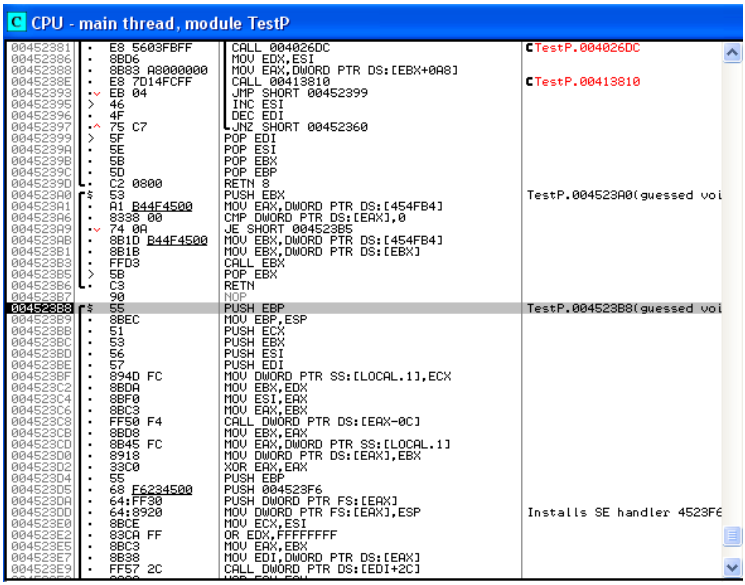


Рисунок 11.19 – Результат роботи програми із заходом у функцію

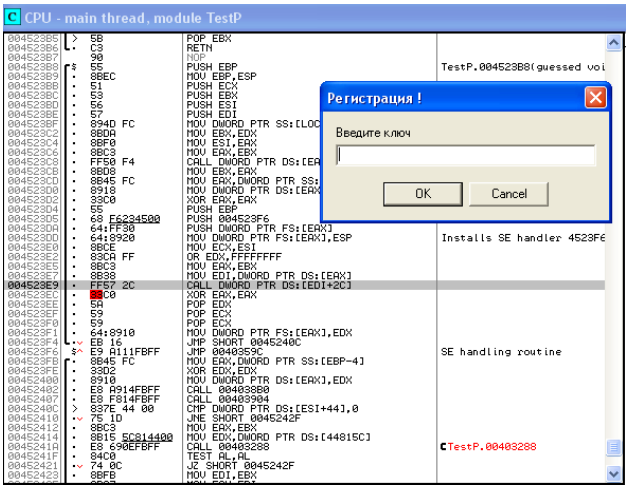


Рисунок 11.20 – Вікно оброблення натиснення на OK

Вдруге перед функцією ставиться друга точка зупинки. Натискаємо Ctrl+F2, після чого виконання зупинилося на першій точці зупинки, а потрібно до другої,

для цього натискаємо ще раз. Виконання дійде до другої точки зупинки. Для заходу у функцію необхідно натиснути клавішу F7 (рис. 11.21).

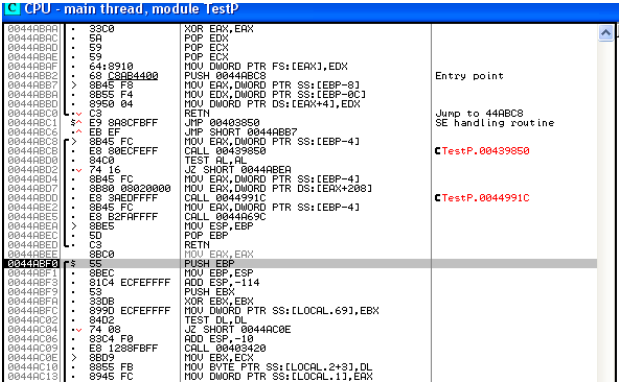


Рисунок 11.21 – Результат заходу до функції

Далі шукаємо чергову функцію. Ставимо третю точку зупинки і заходимо у функцію. Продовжуємо пошук, натискаючи F8. Під час натиснення F8 з'являються API функції (GetCapture, GetActiveWindow). Означає, що незабаром з'явиться діалогове вікно введення ключа (рис. 11.22).

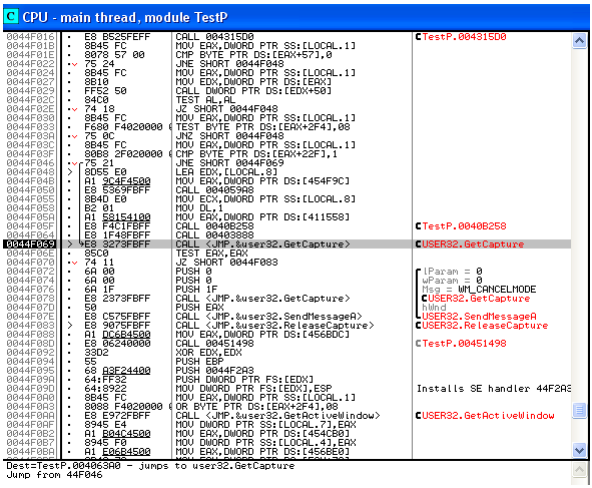


Рисунок 11.22 – Діалогове вікно введення ключа

Натиснення проводиться доти поки з’явиться вікно на панелі завдань, зображене на рис. 11.23, під час трасування програми, яке з’явилося на строчці коду, що надано на рис. 11.24.

Рисунок 11.23 – Вікно на панелі завдань

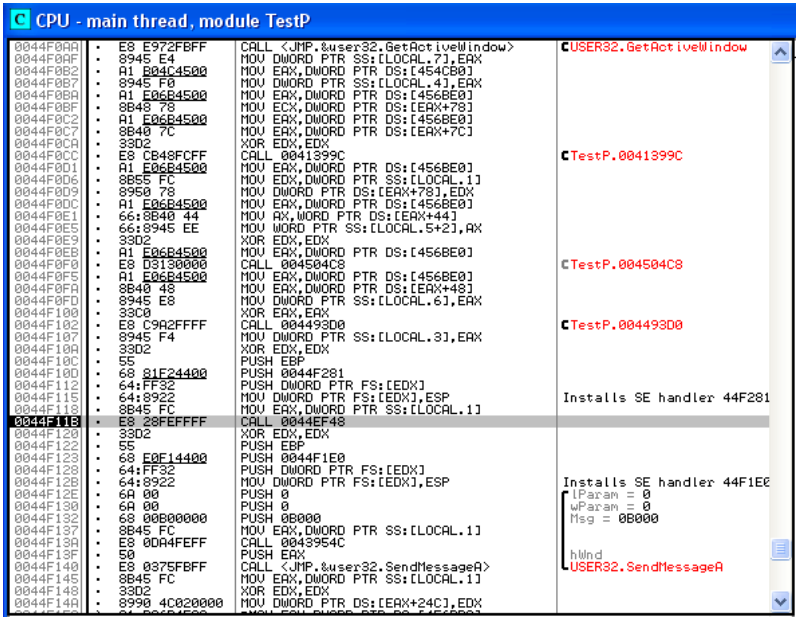


Рисунок 11.24 – Строчка коду, яка викликає вікно на панелі задач

Під час натиснення мишкою на вікно, воно ще не відображається. Це означає, що саме ця функція рисує вікно і десь тут звіряється введений користувачем ключ. Далі продовжується трасування програми (F8) і виявляється, що на цьому місці нескінченний цикл. Ставиться чергова точка зупинки на першому рядку після циклу (MOV DWORD PTR SS:[EBP-8],EAX) (рис. 11.25).

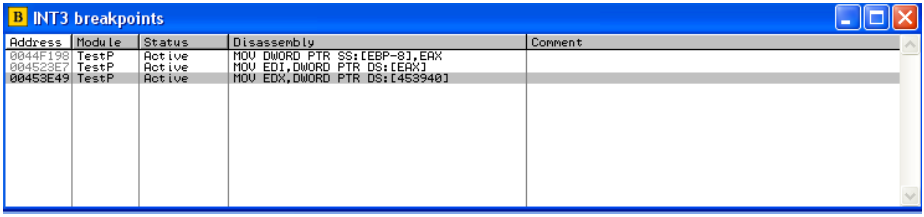


Рисунок 11.25 – Нова точка зупинки

Натискається кнопка , і видно, що віконце повністю відрисовує (рис. 11.26), чекає від нас введення, а програма зупинилася в циклі. Далі вводиться будь-який ключ та натискається OK.

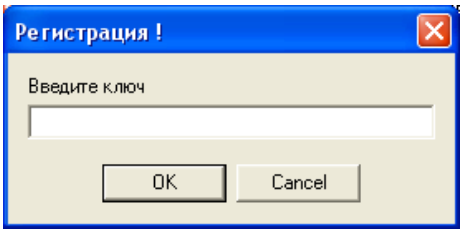


Рисунок 11.26 – Вікно введення ключа

Віконце закривається і відбувається повернення у вікно налагоджувача, а виконання зупиняється на останній точці зупинки. Тепер треба знайти функцію, в якій виводиться вікно з повідомленням, що ключ введений неправильно. Для цього трасується (F8) програма до того моменту, поки не з'явиться необхідне вікно з повідомленням (рис. 11.27).

У лістингу видно рядок ASCII «hello»: це і є **правильний ключ.**

Наступне завдання – зробити так, щоб програма приймала будь-який ключ. Аналізується код перед викликом функції (CALL TestP.00427294), яка виводить вікно з повідомленням про помилку. Програма повинна порівняти введений ключ з яким-небудь ще або перевірити його правильність будь-яким іншим способом. Але для цього вона повинна викликати функцію, яка це все зробить. Шукається

приблизно така структура:

MOV EAX ... //записує в EAX адреси нашого рядка;

CALL ... //виклик функції яка перевірить правильність нашої функції;

JNZ(або **JE**)...// умовний перехід.

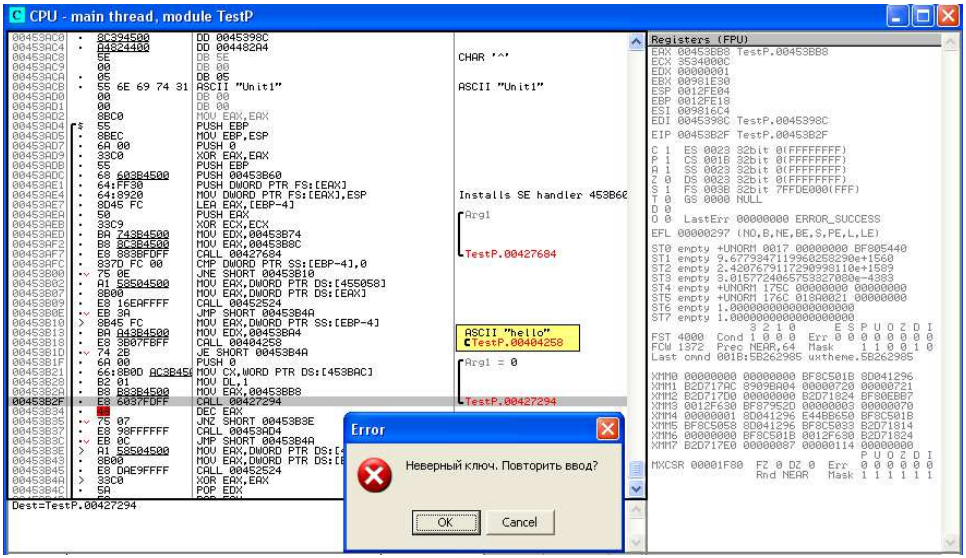


Рисунок 11.27 – Вікно повідомлення «Неправильний ключ»

Умовний перехід. За допомогою цього оператора можна переходити до певного рядка коду, адреса вказується після оператора. Означає: після виклику функції, якщо ключ був введений правильно, то після виклику цього оператора він перейде на рядок, після якого і запуститься правильне виконання програми TESTP.exe. Якщо ж немає, то виконуватиметься код після нього.

При аналізі 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

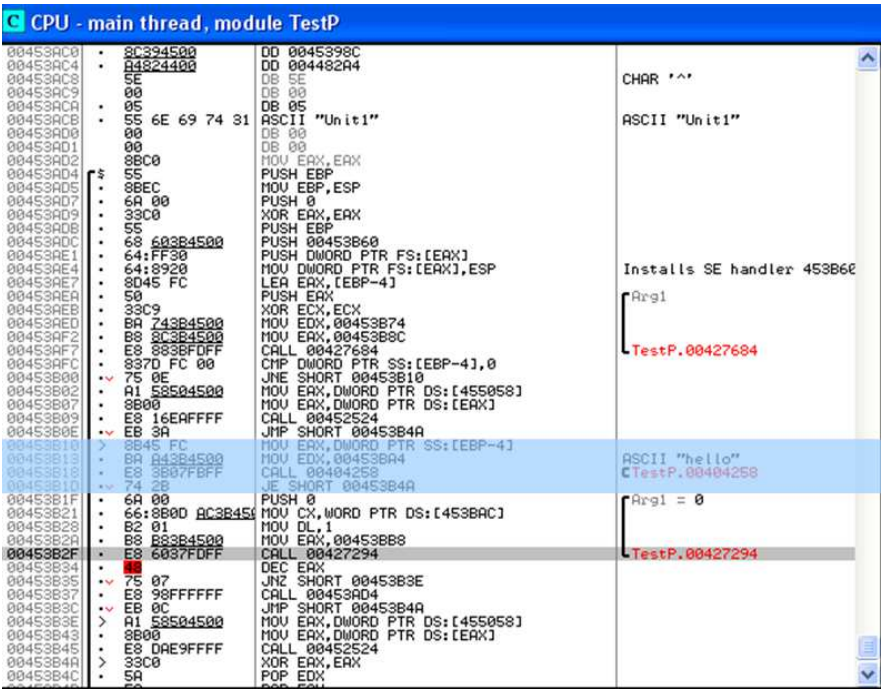


Рисунок 11.28 – Код для аналізу

Тепер наново потрібно покроково виконати цей шматок коду, і стане зрозумілим, що цей оператор **JE SHORT TestP.00453B4A** пропускається і нікуди не перекидає. Це і є функція перевірки правильності введеного значення ключа, і якщо він виявився неправильним, то оператор **JE** нікуди не перевів. Якщо код буде введений правильно, то цей оператор кине нас за адресою 00453B4A. Необхідна адреса зазначена на рис. 11.29 сірим кольором.

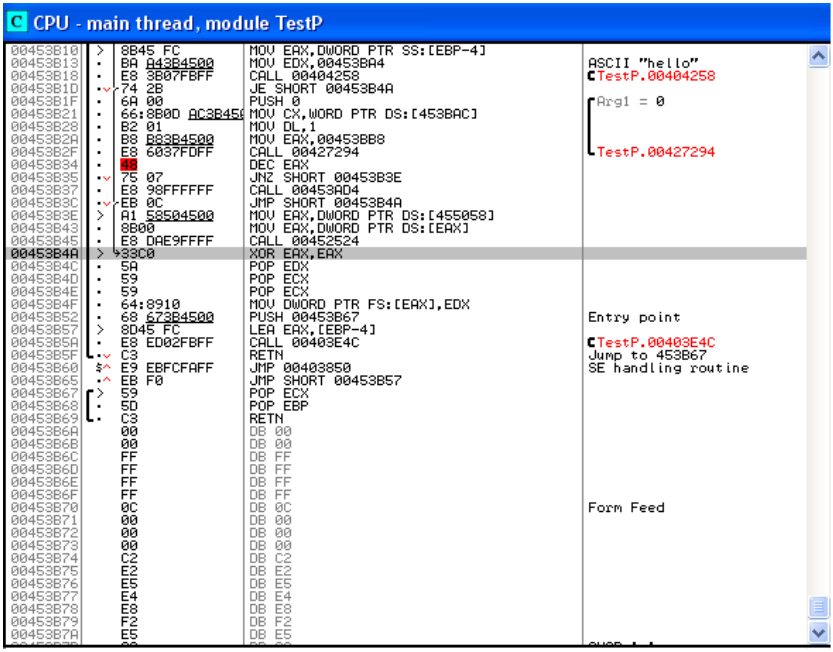


Рисунок 11.29 – Адреса 00453B4A

Виклик функції, яка виводить повідомлення про те, що ключ неправильний, пропускається. Виконується заміна цього оператора на оператора безумовного переходу (**JMP**), який у будь-якому випадку перекине за адресою. Для того щоб замінити рядок, треба виділити та натиснути пропуск (Space), провести заміну JE SHORT 00453B4A (рис. 11.30) на **JMP** SHORT 00453B4A (рис. 11.31).

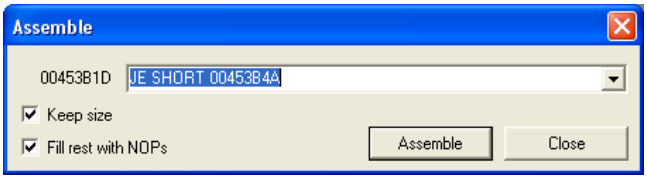


Рисунок 11.30 – Заміна рядка JE SHORT 00453B4A

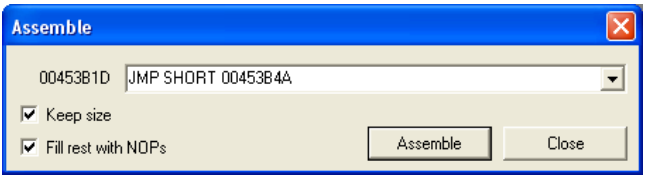


Рисунок 11.31 – Зміна на рядок JMP SHORT 00453B4A

Бачимо, що рядок **JE SHORT TestP.00453B4C** замінився і у віконці **Patches** додався рядок (рис. 11.32).

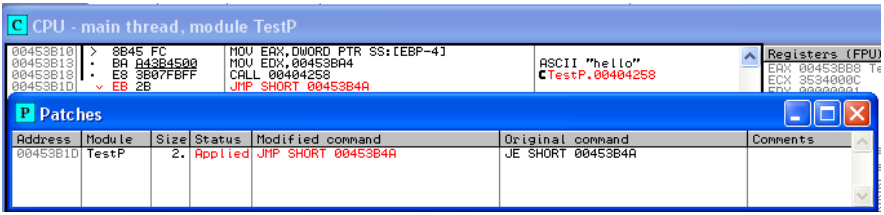


Рисунок 11.32 – Зміни після заміни рядка **JE SHORT TestP.00453B4C**

Програма завжди питатиме 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 простий. Іноді доведеться пробувати по черзі багато інших структур такого типу.

3. Останній крок, який треба зробити, – створити Patch для програми. З цією метою буде задіяний Hex редактор **Xvi32**. Відкрити тестову програму TESTP.exe. З'явиться велика кількість шістнадцяткових цифр, файл розкладений побайтно.

Необхідно знайти рядок JE SHORT TestP.00453B4A в шістнадцятковому вигляді і змінити на JMP SHORT TestP.00453B4A. У вікні CPU (OllyDbg) напроти кожного рядка пишеться її шістнадцятковий вигляд. Для виключення помилки слід узяти одну-дві попередні команди для пошуку, тобто не просто шукати **74 2B**, а ще попередній рядок коду: **E8 3B 07 FB FF 74 2B (рис. 11.33).** Рядок як 74 2B може повторюватися кілька разів, тому вибираємо ще попередній рядок.

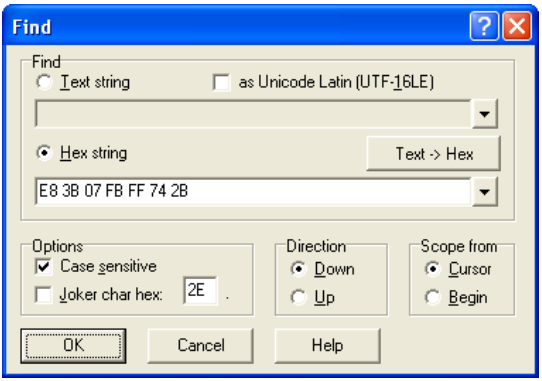


Рисунок 11.33 – Пошук рядка E8 3B 07 FB FF 74 2B

У шістнадцятковому коді виглядає видозмінений рядок **JMP SHORT TestP.00453B4A** (вікно CPU в OllyDbg) так: **EB 2B.** Проводиться заміна, для цього натискається кнопка Replace All (рис. 11.34).

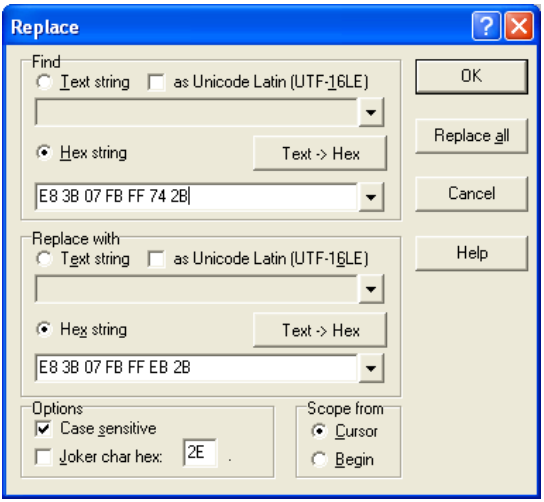


Рисунок 11.34 – Заміна рядків у шістнадцятковому коді

Зберігаються зміни, закривається редактор. Відтепер програма прийматиме будь-який ключ, що і потрібно було зробити.

УСТАНОВЛЕННЯ І НАЛАШТУВАННЯ КРИПТОПРОВАЙДЕРА VIPNET CSP

Отримання і установлення ViPNet CSP

1) Для отримання ViPNet CSP необхідно перейти на офіційний сайт розробника за адресою **http://infotecs.ru/downloads/ product_full.php?id_product=2096** (рис. 11.35, позиція 1) і вибрати дистрибутив ViPNet CSP, відповідний Вашій операційній системі (рис. 11.35, позиція 2).

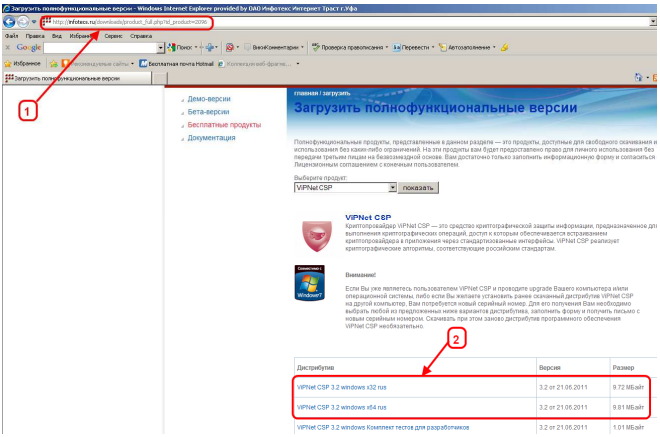


Рисунок 11.35 – Запит на скачування дистрибутива VipNet CSP

2) Пройти процедуру реєстрації, погодившись з умовами ліцензійної угоди (EULA), і заповнити обов'язкові поля (приклад на рис. 11.36, позиції 1–4).

3) Перейти за посиланням (рис. 11.37, позиція 1) для скачування продукту і **зберегти** вказаний серійний номер (рис. 11.37, позиція 2). Скачати продукт можна і з папки рекомендованого програмного забезпечення.

4) Дочекатися закінчення скачування дистрибутива і запустити установку 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) Виконати установлення ViPNet CSP за інструкціями майстра установки.

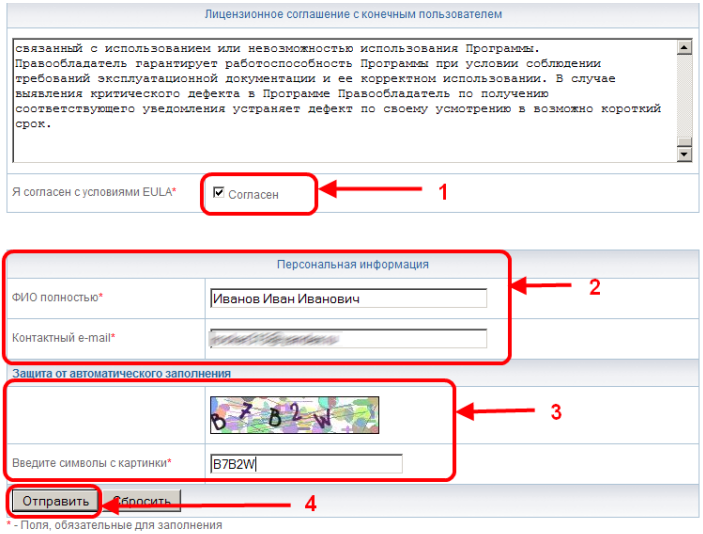


Рисунок 11.36 – Процедура реєстрації VipNet CSP

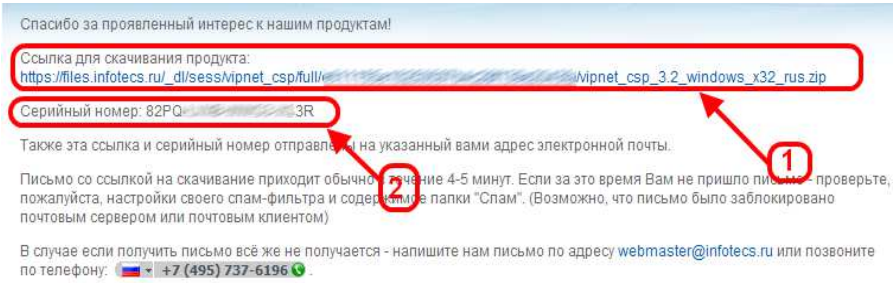


Рисунок 11.37 – Скачування і збереження серійного номера VipNet CSP

6) Після перезавантаження комп’ютера запустити налаштування ViPNet CSP з панелі «Пуск».

7) Зареєструвати ViPNet CSP :

– вибрати «Зареєструвати ViPNet CSP» і натиснути «Далі» (рис. 11.38);

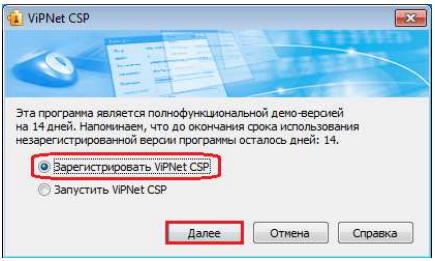


Рисунок 11.38 – Процедура реєстрації VipNet CSP

– вибрати «Запит на реєстрацію (отримати код реєстрації)» і натиснути «Далі» (рис. 11.39);

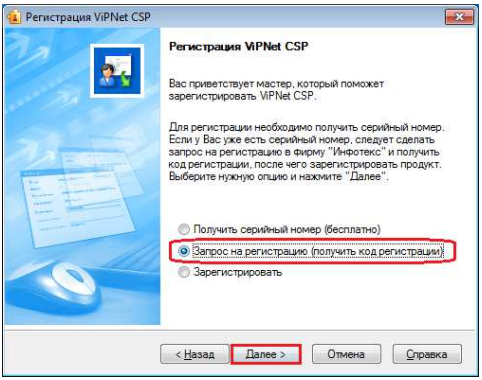


Рисунок 11.39 – Процедура реєстрації VipNet CSP

– вибрати «Через Інтернет (online)» і натиснути «Далі» (рис. 11.40);

– заповнити форму своїми реєстраційними даними, включаючи «Серійний номер» ViPNet CSP, отриманий при реєстрації на кроці 3 (рис. 11.37) (так само серійний номер можна знайти на Вашому E-mail, вказаний при завантаженні ViPNet CSP з web сайту) і натиснути «Далі» (рис. 11.41);

– запустити процес запиту на реєстрацію (рис. 11.42).

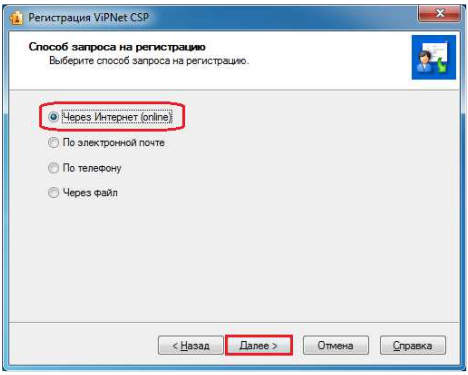


Рисунок 11.40 – Процедура реєстрації VipNet CSP

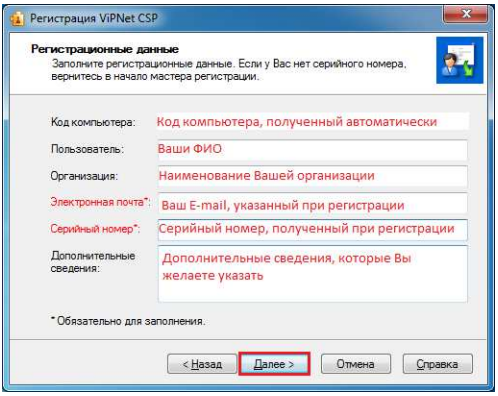


Рисунок 11.41 – Процедура реєстрації даних VipNet CSP

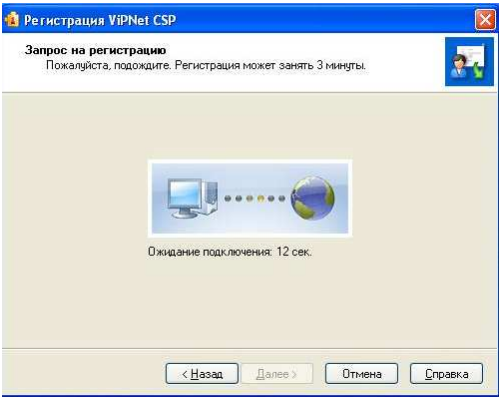


Рисунок 11.42 – Процедура запиту реєстрації

Увага! Якщо при виконанні дії, вказаної на даному кроці при натисненні на кнопку «Далі» (рис. 11.41), процес очікування виконання запиту займає більш ніж 3 хвилини і вилітає повідомлення, показане на рис. 11.43.

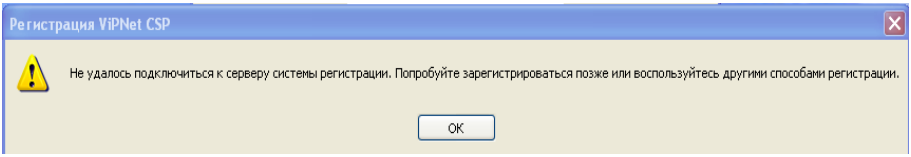


Рисунок 11.43 – Помилка при реєстрації VipNet CSP

У цьому випадку необхідно:

a) переконатися в доступності мережі Інтернет. Якщо Інтернет не доступний, то необхідно усунути причину відмови, звернувшись до системного адміністратора або в компанію, що надає доступ до мережі Інтернет;

b) якщо Інтернет доступний, повернутися у вікно вибору способу реєстрації (рис. 11.41) і скористатися альтернативними способами реєстрації: «Електронною поштою» або «За телефоном», слідуючи інструкціям майстра реєстрації;

c) якщо Інтернет доступний, але вилітає повідомлення про помилку, то необхідно встановити ViPNet Csp (поверх вже встановленої) спеціальної версії:

http://files.iitrust.ru/csp/3.2.(4.8517)/x32/Setup.exe – для 32 розрядної операційної системи Windows;

http://files.iitrust.ru/csp/3.2.(4.8517)/x64/Setup.exe – для 64 розрядної операційної системи Windows;

Наново зробити запит (online) (рис. 11.41–11.42).

У разі успішної реєстрації натисніть «Готово» (рис. 11.44).

На запитання «Запустити ViPNet CSP зараз?» відповісти «Yes» або запустити ViPNet CSP пізніше з панелі «Пуск».

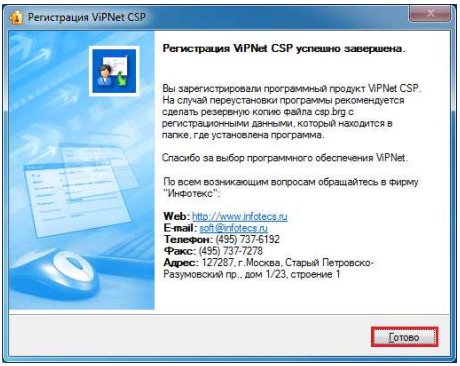


Рисунок 11.44 – Успішна реєстрація VipNet CSP

Налаштування VipNet CSP для роботи з електронним цифровим підписом

Установлення особистого сертифікату

− Скопіюйте Ваші облікові дані (папка з Вашими «Прізвище Ім’я По батькові») з компакт-диска (або з іншого зовнішнього носія) в будь-яке місце жорсткого диска комп’ютера.

− З головного вікна ViPNet CSP перейдіть на закладку «Контейнери», як зображено на рис. 11.45.

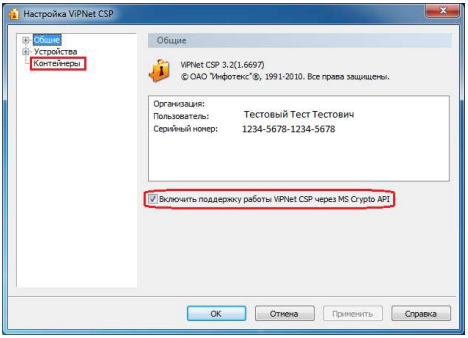


Рисунок 11.45 – Головне вікно VipNet CSP

− Натисніть «Додати», вкажіть місцеположення ключів (папка «Прізвище Ім’я По батькові»), виберіть «Ім’я контейнера» вигляду «XXXXXXXX-XXXX-

XXXX-XXXX-XXXXXXXXXXXX» (єдиний файл у випадному списку без розширення) і натисніть «OK» (рис. 11.46–47).

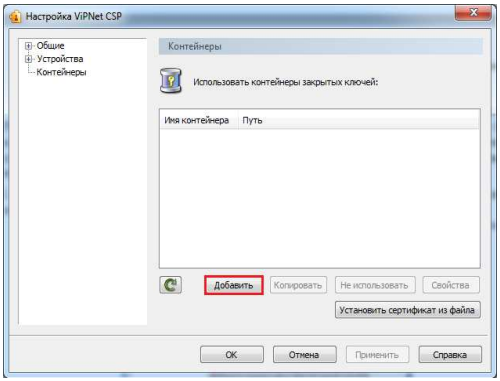


Рисунок 11.46 – Налаштування ViPNet CSP

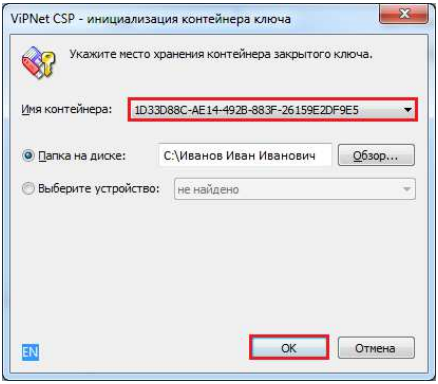


Рисунок 11.47 – Ініціалізація контейнера ключа

– ViPNet CSP видасть повідомлення «Контейнер успішно доданий» і поставить питання про установлення знайдених у контейнері сертифікатів у системне сховище. Натисніть «Да» і перейдіть до наступного пункту інструкції (рис. 11.48).

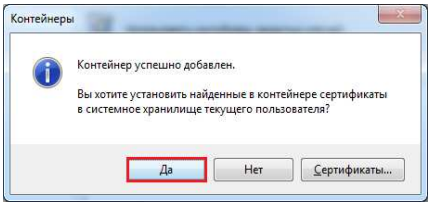


Рисунок 11.48 – Повідомлення «Контейнер успішно доданий»15

Увага! Рекомендується змінити пароль доступу до контейнера із стандартного 123456 на стійкіший, який знатимете тільки Ви. Для цього на вкладці «Контейнери» виділите Ваш контейнер, натисніть «Властивості» «→Змінити пароль» (рис. 11.49–11.50) і йдіть за інструкціями Майстра.

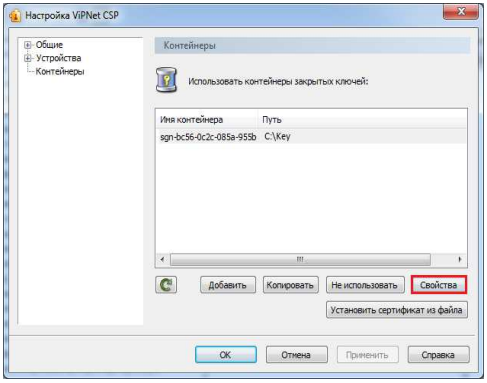


Рисунок 11.49 – Налаштування ViPNet CSP

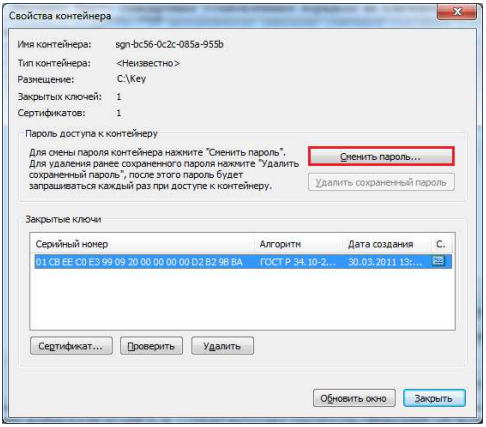


Рисунок 11.50 – Властивості контейнера

Установлення кореневого сертифікату і списків відгуку сертифікатів Акредитованого центру

1) В Інтернет оглядачі відкрити посилання http://iitrust.ru/services/uc/sert.php (рис. 11.51, позиція 1).

2) Знайти рядок «УЦ ИИТ (К2) (XXXX) (82)» (рис. 11.51, позиція А).

3) Скачати Кореневий сертифікат УЦ (рис. 11.51, позиція 2) і Списки відкликаних сертифікатів (рис. 11.51, позиція 3).

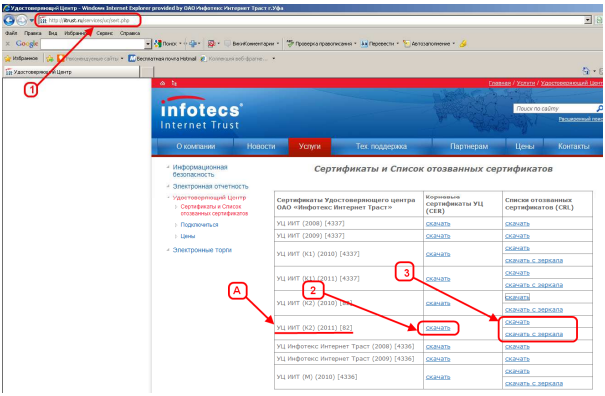


Рисунок 11.51 – Посилання на скачування кореневого сертифіката

Завантаження і установлення кореневого сертифіката:

При відкритті посилання (див. рис. 11.51, позиція 2) відкрити кореневий сертифікат безпосередньо з вікна Web-браузера, натиснувши «Відкрити» (рис. 11.52).

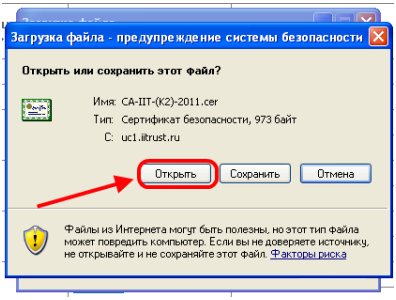


Рисунок 11.52 – Завантаження кореневого сертифіката

Натиснути «Встановити» (рис. 11.53).

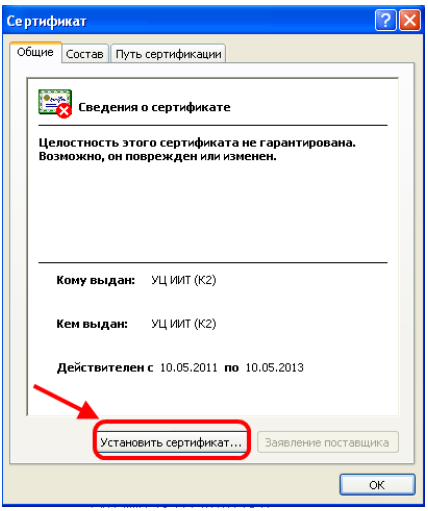


Рисунок 11.53 – Установлення кореневого сертифіката

У майстері імпорту сертифікатів Windows натиснути «Далі» (рис. 11.54).

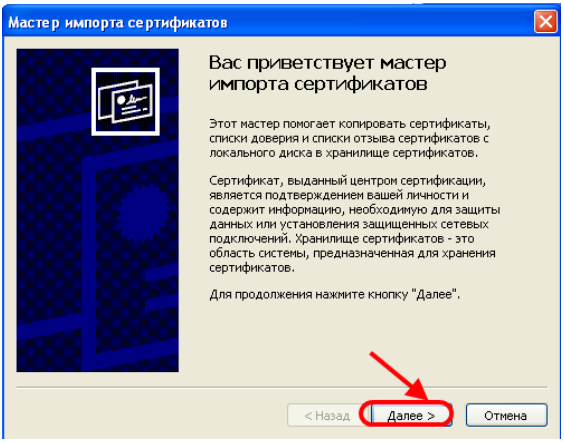


Рисунок 11.54 – Майстер імпорту сертифіката

Вибрати «Помістити всі сертифікати в наступне сховище» (рис. 11.55, позиція 1). Натиснути «Огляд» (рис. 11.55, позиція 2). Вибрати «Довірені кореневі центри сертифікації» (рис. 11.55, позиція 3). Натиснути «Ок» (рис.11. 55, позиція 4).

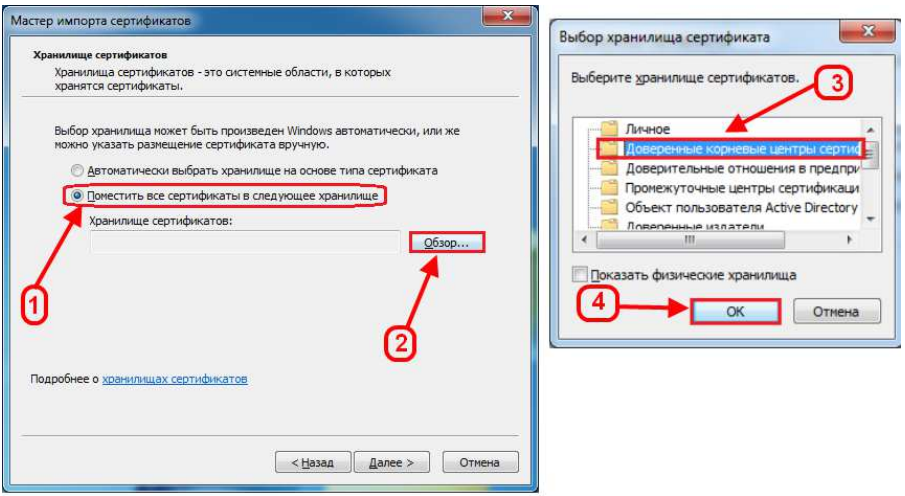


Рисунок 11.55 – Приміщення всіх сертифікатів у сховищі

У вікні попередження системи безпеки натиснути «Так» (рис. 11.56).

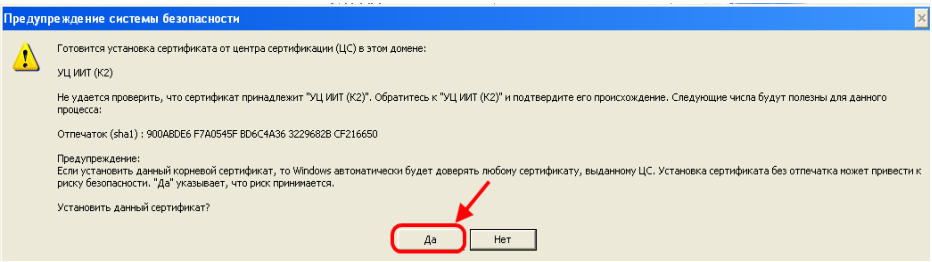


Рисунок 11.56 – Попередження системи безпеки

При успішному імпорті кореневого сертифіката, з’явиться сповіщення: натиснути «ОК» (рис. 11.57).

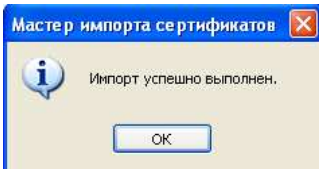


Рисунок 11.57 – Успішний імпорт кореневого сертифіката

Закрити вікно кореневого сертифіката (рис. 11.58).

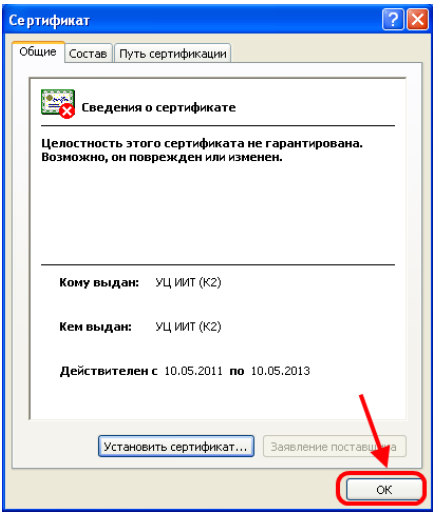


Рисунок 11.58 – Вікно кореневого сертифіката

Завантаження і установлення списку відгуку сертифікатів УЦ:

Після відкриття посилання (рис. 11.51, позиція 3) натиснути «Зберегти» (рис. 11.59).

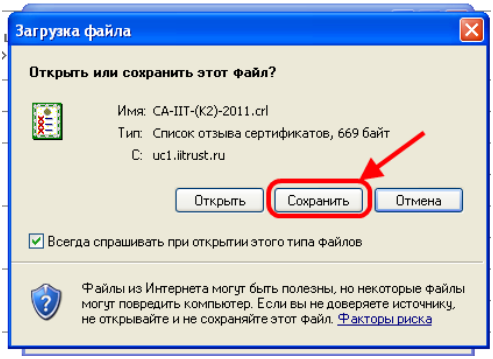


Рисунок 11.59 – Установлення списку відгуку сертифікатів

Зберегти список відгуку сертифікатів, наприклад, на робочому столі (рис. 11.60, позиції 1–2).

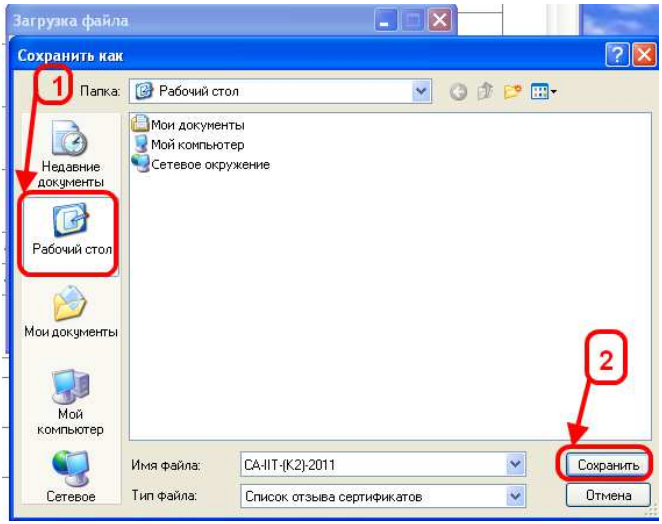


Рисунок 11.60 – Установлення списку відгуку сертифікатів

Закрити вікно після завершення завантаження (рис. 11.61).

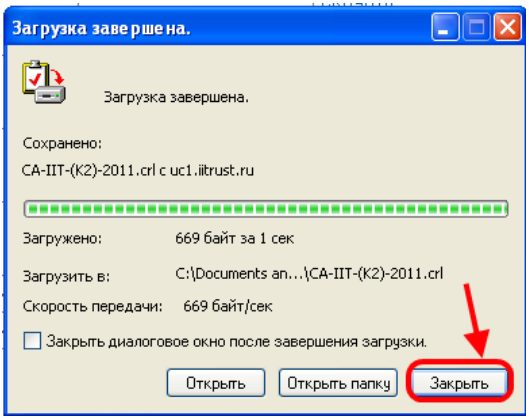


Рисунок 11.61 – Завершення завантаження списку відгуку сертифікатів

Виділити завантажений і збережений на робочому столі список відгуку сертифікатів правою кнопкою мишки (рис. 11.62, позиція 1) і з випадного меню вибрати «Установити список відгуку (CRL)» (рис. 11.62, позиція 2).

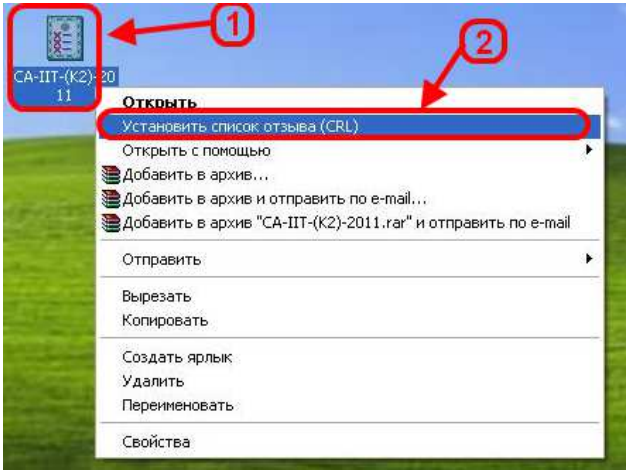


Рисунок 11.62 – Установлення списку відгуку сертифікатів

У майстрі імпорту сертифікатів Windows натиснути «Далі» (рис. 11.63).

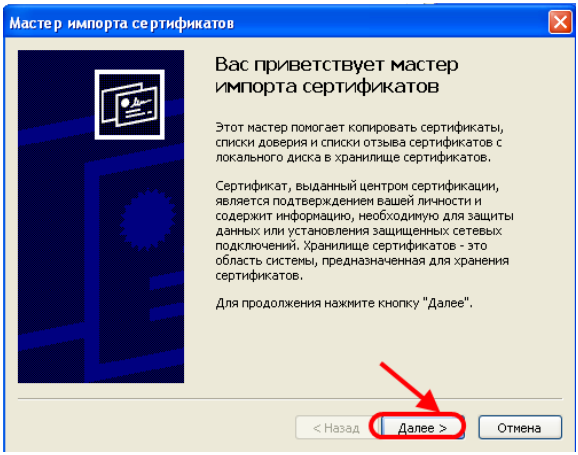


Рисунок 11.63 – Майстер імпорту сертифікатів

Вибрати «Помістити всі сертифікати в наступне сховище» (рис. 11.64, позиція 1). Натиснути «Огляд» (рис. 11.64, позиція 2). Вибрати «Проміжні центри сертифікації» (рис. 11.64, позиція 3). Натиснути «Ок» (рис. 11.64, позиція 4).

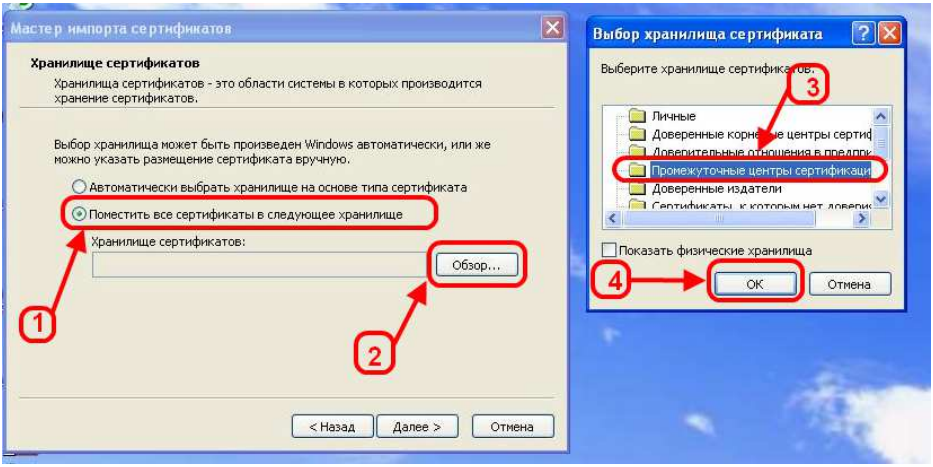


Рисунок 11.64 – Приміщення сертифікатів у сховищі

У результаті успішного імпорту кореневого сертифіката, з'явиться сповіщення: натиснути «ОК» (рис. 11.65).

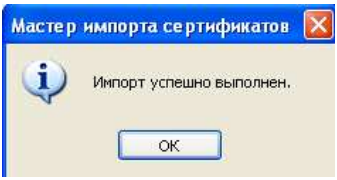


Рисунок 11.65 – Успішне завершення установлення списку відгуку сертифікатів

На цьому процедуру установлення кореневого сертифіката і списків відгуку сертифікатів Акредитованого центру завершено.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Перед початком виконання роботи встановити ВСЕ задане програмне забезпечення

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

ВАРІАНТ 1

1. Cast (128), Triple DES (192), Serpent (128).
2. Enc-01.txt.
3. Варіант-01.rar
4. Проводити злом пароля, яким зашифрований файл, при заданих фіксованих параметрах (вибрати самостійно і відобразити в додатку), наприклад, з довжиною пароля, рівною 8 символам, алгоритмом хешування CALG_SHA і так далі.

ВАРІАНТ 2

1. Triple DES (192), Serpent (128), Skipjack.

2. Enc-02.txt.

3. Варіант-02.zip

4. Розробити діалогове вікно задання параметрів цифровому підпису.

ВАРІАНТ 3

1. Serpent (128), Skipjack, AES-256.

2. Enc-03.txt.

3. Варіант-03.rar

4. Дослідити параметри криптопровайдерів, установлених на комп'ютері.

ВАРІАНТ 4

1. Vigenure, Enigma, RC4.

2. Enc-04.txt.

3. Варіант-04.zip

4. Перерахувати всі криптопровайдери, встановлені на комп'ютері.

ВАРІАНТ 5

1. Enigma, RC4, SEED.

2. Enc-05.txt.

3. Варіант-05.rar

4. Побудувати 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. Дослідити 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. Проводити обмін сесійними ключами з іншим екземпляром додатка за іменованим каналом і відображенням отриманого ключа.

ВАРІАНТ 8

1. AES-256, Substitution, Cobra128 .

2. Enc-08.txt.

3. Варіант-08.zip

4. Побудувати тестуючу процедуру, початковими даними якої був би цифровий підпис, а процедура перевіряла відповідно до встановлених параметрів правильність даного підпису. За наявності помилки по можливості виправляти її.

ВАРІАНТ 9

1. Substitution, Cobra128, DES.

2. Enc-09.txt.

3. Варіант-09.rar

4. Досліджувати не менше 4 алгоритмів хешування і подати результати порівняння в клієнтській ділянці вікна.

ВАРІАНТ 10

1. DESX, Scytale, RC2.

2. Enc-10.txt.

3. Варіант-10.zip

4. Розробити процедуру верифікації (перевірки) сертифіката відкритого ключа (PKI), що міститься у файлі *.cer і надати результати такої перевірки.

ВАРІАНТ 11

1. MARS, Twofish (128), IDEA.

2. Enc-11.txt.

3. Варіант-11.rar

4. Розробити додаток, який на основі даних з системного реєстру виводить в клієнтську ділянку і у файл на диску список усіх криптопровайдерів, з описом використовуваних ними алгоритмів, довжин ключів (для хеш-функцій – розмірів блоків), ідентифікаторів цих алгоритмів.

ВАРІАНТ 12

1. Cobra128, DES, Vigenure.

2. Enc-12.txt.

3. Варіант-12.zip

4. Створити цифровий підпис у форматі PKSC#7 за допомогою функції CryptSignMessage() шляхом відкриття сховища сертифікатів (CertOpenStore()) і вибору потрібного сертифіката (CertFindCertificateInStore()).

ВАРІАНТ 13

1. RC4, SEED, Cast (256).

2. Enc-13.txt.

3. Варіант-13.rar

4. Організувати отримання ідентифікатора безпеки поточного користувача.

ВАРІАНТ 14

1. DES, Vigenure, Enigma.

2. Enc-14.txt.

3. Варіант-14.zip

4. Розробити програму, в якій ідентифікатор безпеки перетвориться в рядкове подання, використовуючи функцію ConvertSidToStringSid. Задати графічний інтерфейс для програми.

ВАРІАНТ 15

1. SEED, Cast (256), DESX.

2. Enc-15.txt.

3. Варіант-15.rar

4. Розробити програму, з ініціалізацією ідентифікатора безпеки, використовуючи функцію initiaiizesid. Заздалегідь функцією GetSidLengthRequired визначати довжину ідентифікатора безпеки. Після ініціалізації ідентифікатора безпеки за допомогою функції GetSidSubAuthority визначити індекс відносного ідентифікатора, після чого встановити його значення. Потім перевірити 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. Дослідити привілеї власного облікового запису користувача. Надати можливість установлення не менше 10 різних привілеїв

ВАРІАНТ 17

1. Twofish (256), Twofish (128), RC6.

2. Enc-17.txt.

3. Варіант-17.rar

4. Розробити 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. Розробити додаток, який основні дані з системного реєстру виводить в клієнтську ділянку у файл на диску список усіх криптопровайдерів, з описом використовуваних ними алгоритмів, довжин ключів (для хеш-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. Розробити програму, яка визначає ім’я облікового запису і 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. Розробити програму, яка визначає ідентифікатор безпеки і ім’я домену за іменем облікового запису, використовуючи функцію LookupAccountName.

ВАРІАНТ 21

1. Rail Fence Algoritm, RC6, Serpent (256).

2. Enc-21.txt.

3. Варіант-21.rar

4. Розробити програму, в якій ініціалізується дескриптор безпеки для нового каталогу. При цьому власник каталогу і первинна група власника каталогу визначаються операційною системою, використовуючи механізм, заданий за умовчанням. У цьому випадку власником каталогу є користувач, від імені якого запущена програма. Оскільки списки управління доступом створюються порожніми, то доступ до каталогу дозволений всім користувачам.

ВАРІАНТ 22

1. RC5, Caesar, Camelia.
2. Enc-22.txt.
3. Варіант-22.zip
4. Розробити діалогове вікно задання параметрів цифровому підпису.

ВАРІАНТ 23

1. IDEA, SHAKAL-2, AES-192.
2. Enc-23.txt.
3. Варіант-23.rar
4. Дослідити параметри криптопровайдерів, установлених на комп'ютері.

ВАРІАНТ 24

1. Caesar, Camelia, Cast (128).
2. Enc-24.txt.
3. Варіант-24.zip
4. Перерахувати всі криптопровайдери, встановлені на комп'ютері.

ВАРІАНТ 25

1. SHAKAL-2, AES-192, MARS.
2. Enc-25.txt.
3. Варіант-25.rar

4. Побудувати 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. Дослідити параметри вибраного користувачем криптопровайдера, встановленого на комп'ютері.

ВАРІАНТ 27

1. RC6, Serpent (256), Affine Shift.

2. Enc-27.txt.

3. Варіант-27.rar

4. Провести обмін сесійними ключами з іншим екземпляром додатка за іменованим каналом і відобразити отриманий ключ.

ВАРІАНТ 28

1. AES -128, Twofish (256), Rail Fence Algoritm.

2. Enc-28.txt.

3. Варіант-28.zip

4. Побудувати тестуючу процедуру, початковими даними якої був би цифровий підпис, а процедура перевіряла б відповідно до встановлених параметрів правильність даного підпису. За наявності помилки по можливості виправляти її.

ВАРІАНТ 29

1. ARCFOUR, Blowfish (448), RC5.

2. Enc-29.txt.

3. Варіант-29.rar

4. Дослідити не менше 4 алгоритмів хешування і подати результати порівняння в клієнтській ділянки вікна.

ВАРІАНТ 30

1. Blowfish (448), RC5, Caesar.
2. Enc-30.txt.
3. Варіант-30.zip
4. Організувати отримання ідентифікатора безпеки поточного користувача.

Хід роботи

1. Дослідницька частина

Етап 1. Шифрування текстового файлу

За допомогою блокнота підготувати текстовий файл Test.txt, який повинен містити прізвище, ім'я і по батькові студента, назву факультету і номер групи. *Наприклад, Коваленко Микола Миколайович, Економічної інформатики, група. 6.04.51.13.01.* Використовуючи заданий в **п.1** індивідуального завдання алгоритм шифрування і його програмну реалізацію з описаних у цьому керівництві, провести шифрування. Підрахувати контрольні суми кожного файлу, що підтверджують його цілісність, і занести результати до таблиці 11.2. Провести підтвердження результатів шифрування відповідними скриншотами.

Таблиця 11.2 – Результати шифрування текстового файлу

№ п/п	Ім'я файлу	Алгоритм шифрування	Пароль (ключ)	Розмір файлу, байт	Контрольна сума	Пояснення
1	Test.txt	–	–	78	ATSM-0MTH	Початковий файл
2						Зашифровані файли
3						
4						

Етап 2. Дешифрування текстового файлу

Аналіз викраденого текстового файлу, заданого в **п.2.** індивідуального завдання, показав, що він зашифрований. В ході спілкування з його автором вдалося з'ясувати тільки те, що при шифруванні використовувався простий пароль: *qwerty*. Задані в роботі програмні засоби ДОСИТЬ для розтину текстового файлу.

Провести підтвердження результатів дешифрування відповідними скриншотами. Обов'язково вказати назву програми, за допомогою якої вдалося дешифрувати і алгоритму.

2. Аналітична частина.

Етап 3. Злом парольного захисту файлу

У результаті закачування з Інтернет архівного файлу, заданого в **п.3** індивідуального завдання, було з'ясовано, що він закритий паролем. Необхідно зламати пароль, враховуючи, що він схожий на PIN-коди карток VISA службовців.

Використовуючи додаткову інформацію, вдалося встановити, що для документа office (.pdf) користувач у вигляді пароля задав номер телефону. Необхідно визначити пароль.

Необхідно зламати документ і подати в звіті обов'язкове значення паролів, вивести вміст файлів і провести підтвердження результатів злому відповідними скриншотами.

Етап 4. Виконання крекінгу тестового виконуваного файлу TestP.exe

Провести зміну базового тестового пароля, який прописаний усередині файлу TESTP.exe на значення «kaixo» (6B 61 69 78 6F), а також зробити так, щоб при введенні правильного ключа видавалося повідомлення про помилку, а при введенні будь-якого іншого (неправильного) ключа – відбувалася правильна умовна реєстрація програми.

Супроводжувати процес крекінгу відображенням скриншотів, з обов'язковим поданням підсумкового вигляду вікон Breakpoints і Patches програми OllyDbg.

Етап 5. Отримання і реєстрація сертифіката електронного цифрового підпису

Провести установку криптопровайдера центру сертифікації на свій комп'ютер. Отримати 2 сертифікати:

1. Кореневий сертифікат, що засвідчує центр ВАТ «Інфотекс Інтернет Траст».

2. Створити власний сертифікат, що засвідчує центр ВАТ «Інфотекс Інтернет Траст».

У звіті подати скриншоти отриманих сертифікатів.

ДОДАТКОВЕ ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

Створити додаток, в якому реалізується за допомогою CryptoAPI індивідуальне завдання, визначене в **п.4.**

Контрольні запитання

1. Що таке сесійний ключ?
2. Поясніть порядок отримання сесійного ключа.
3. У чому специфіка підпису документів?
4. З якою метою використовується цифровий електронний підпис?
5. Чим відрізняється закритий ключ від сесійного ключа?
6. Наведіть приклади криптопровайдерів.
7. Опишіть процес створення криптопровайдера.

Лабораторная робота 12

ДОСЛІДЖЕННЯ І ОПТИМІЗАЦІЯ ЗАВАНТАЖЕННЯ ОС WINDOWS

Мета роботи. Ознайомитися з етапами і кроками завантаження і завершення роботи операційної системи Windows на основі BIOS, особливостями і порядком переходу на технологію завантаження UEFI, з аналізом аварійних дампів під час збоїв і помилок в роботі ОС.

Указівки з підготовки до виконання лабораторної роботи

Необхідно вивчити лекційний матеріал та вказану додаткову літературу:

1. Руссинович М. Внутреннее устройство Microsoft Windows. Основные подсистемы ОС / М. Руссинович, Д. Соломон, А. Ионеську – Спб.: Питер, 2014. – С. 568–671.

2. Таненбаум Е. Современные операционные системи. / Е. Таненбаум – Спб.: Питер, 2010. – С. 967–968.

Теоретичні відомості

Ще з часів Windows NT завантажувачем операційної системи був **NTLDR**, а починаючи з Windows Vista, він замінений 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), для заміни базової системи 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 з пристрою завантаження перший сектор, який є головним завантажувальним записом **MBR** (**Master Boot Record).** Перед записом завантажувального сектора програма Setup повинна визначити формат розділу, оскільки від цього залежатиме вміст цього сектора. Для розділу, який відформатований під NTFS, Windows записує код NTFScapable. Цей код надає Windows інформацію про структуру і формат диска і здійснює читання з файлу

BOOTMGR в кореневому каталозі тома. Після того як файл BOOTMGR опиниться в пам'яті, код NTFScapable передає управління його точці входу.

Диспетчер завантаження BOOTMGR є файлом невеликого розміру, розташованим в кореневому каталозі активного розділу. Основне його призначення – забезпечення подальшої процедури завантаження відповідно до існуючої **конфігурації**, що зберігається в спеціальному **сховищі даних конфігурації** (**BCD - Boot Configuratin Data)**, що є файлом з ім'ям **BCD** та знаходиться в каталозі **BOOT** активного розділу.

Завантаження Windows з диспетчером BOOTMGR, активний розділ, як мінімум, повинен 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 випадку із завантаженням Windows диспетчер **bootmgr** прочитує зі сховища конфігураційні дані, необхідні для завантаження ядра системи, і передає управління додатку, що виконує наступний етап (**winload.exe**).

Структура файлу **\BOOT\BCD** є кущем реєстру і відображається в редакторові реєстру Windows як **HKEY_LOCAL_MACHINE\BCD000000** (рис. 12.1).

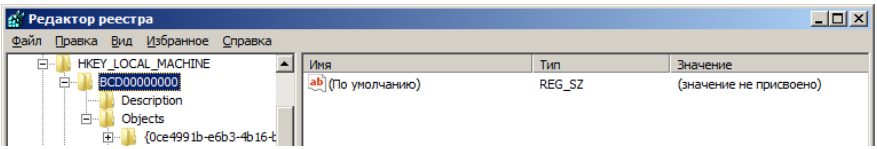


Рисунок 12.1 – Файл **\BOOT\BCD** у редакторі реєстру

Розділ конфігурації BCD містить підрозділ **Description** з параметрами опису і підрозділ **Objects** з об'єктами конфігурації завантаження. Дані конфігурації завантаження можна умовно розділити на 3 основних складових:

- сховище BCD (Store);
- записи в сховищі (Entries);
- параметри записів (Entry Options).

Ієрархічно сховищем конфігурації завантаження є сукупність об’єктів (Objects), що складаються з окремих елементів (Elements) (рис. 12.2).

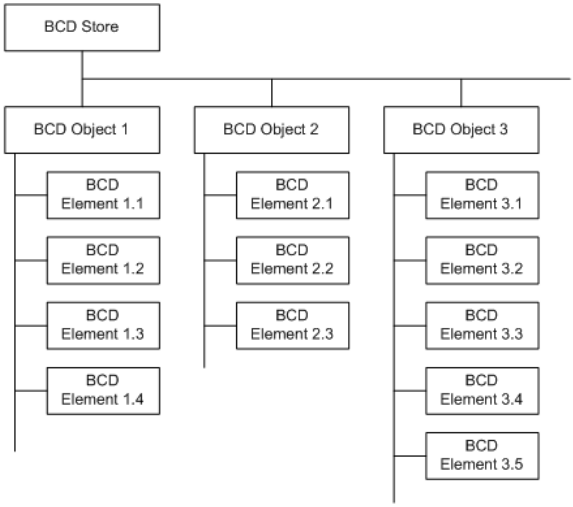


Рисунок 12.2 – Сховище конфігурації

Кожен з об’єктів є впорядкованою структурою елементів, що обробляється диспетчером завантаження. Кожен об’єкт BCD має ім’я, яке є **глобальним унікальним ідентифікатором – GUID**. Ідентифікатор GUID формується програмним шляхом і однозначно є унікальним для тієї системи, де він створюється. Позначається GUID у вигляді груп з шістнадцяткових цифр, що розділяються дефісами, і розміщеним у фігурних дужках (табл. 12.1). Існує 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 зв’язують деякі з ідентифікаторів GUID з внутрішніми ідентифікаторами (псевдонімами) редактора **bcdedit.**

Таблиця 12.1 – Ідентифікатор GUID

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 з розділів підрозділу **Objects** також складається з двох підрозділів – **Descriptions** з описом типу об’єкта і **Elements**, що визначає набір елементів з параметрами об’єкта. У розділі **Description** є ключ **Type** типу **REG_DWORD**, значення розрядів 28–31 якого визначає тип об’єкта (значення в старшій тетраді старшого байта):

– **1********* – додаток;

– **2********* – успадкований об’єкт;

– **3********* – пристрій.

Команда **bcdedit /create** передбачає тільки наступні комбінації:

1. Об’єкти додатка наведені в таблиці 12.2.

Таблиця 12.2 – Об’єкти додатку

Type	Параметри створення BCDEDIT
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

2. Успадковані об’єкти наведені в таблиці 12.3.

Таблиця 12.3 – Об’єкти додатка

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}

Закінчення табл. 12.3

Type	Параметри створення BCDEDIT
20200005	/create /inherit memdiag
20200006	/create /inherit ntldr
20200007	/create /inherit setupldr
20200008	/create /inherit bootsector
20200009	/create /inherit startup
20300000	/create /inherit device

3. Об’єкти пристроїв наведені в таблиці 12.4.

Таблиця 12.4 – Об’єкти пристроїв

Type	Параметри створення BCDEDIT
30000000	/create /device /create {ramdiskoptions}

Кожен BCD-елемент має свій власний ключ реєстру. Назва ключа подана 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**. Ім’я ключа пов’язане з його вмістом, так наприклад, підрозділ з ім’ям **12000004** завжди містить строковий параметр з текстовим описом елемента:

[HKEY_LOCAL_MACHINE\BCD00000000\Objects\{b2721d73-1db4-4c62-bf78-c548a880142d}\Elements\12000004]

"Element"="Диагностика пам’яті"

У таблиці 12.5 подані **BCD-**елементи, які відомі в декількох значеннях. Так, у першому стовпці показаний 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	Имя элемента	Константа	Имя элемента	Константа	Имя элемента
11000001	device	22000011	kernel	26000001	pxesoftreboot
12000002	path	22000012	hal	26000003	customsettings
12000004	description	22000013	dbgtransport	26000004	stampdisks
12000005	locale	22000023	bcdfilepath	26000004	pae
12000016	targetname	22000053	evstore	26000005	resume
12000019	busparams	220000F1	hypervisorpath	26000005	cacheenable
12000030	loadoptions	23000003	default	26000006	debugoptionenabled
1200004A	fontpath	23000003	resumeobject	26000010	detecthal
14000006	inherit	23000006	resumeobject	26000020	displaybootmenu
14000008	recoverysequence	24000001	displayorder	26000021	noerrordisplay
15000007	truncatememory	24000002	bootsequence	26000022	winpe
1500000B	integrityservices	24000010	toolsdisplayorder	26000024	nocrashautoreboot
1500000C	firstmegabytepolicy	25000001	passcount	26000025	lastknowngood
1500000D	relocatephysical	25000002	testmix	26000026	oslnointegritychecks
1500000E	avoidlowmemory	25000003	failurecount	26000027	osltestsigning
15000011	debugtype	25000004	timeout	26000030	nolowmem
15000012	debugaddress	25000004	testtofail	26000040	vga
15000013	debugport	25000007	bootux	26000041	quietboot
15000014	baudrate	25000020	nx	26000042	novesa
15000015	channel	25000021	pae	26000051	usephysicaldestination
15000018	debugstart	25000031	removememory	26000054	uselegacyapicmode
15000022	emsport	25000032	increaseuserva	26000060	onecpu
15000023	emsbaudrate	25000033	perfmem	26000062	maxproc
15000042	keyringaddress	25000050	clustermodeaddressing	26000064	maxgroup
15000047	configaccesspolicy	25000052	restrictapiccluster	26000065	groupaware
15000051	initialconsoleinput	25000061	numproc	26000070	usefirmwarepcisettings
15000052	graphicsresolution	25000063	configflags	26000081	safebootalternateshell
16000009	recoveryenabled	25000066	groupsize	26000090	bootlog
1600000B	badmemoryaccess	25000071	msi	26000091	sos
1600000F	traditionalksegmappings	25000072	pciexpress	260000A0	debug

Закінчення табл. 12.5

Константа	Имя элемента	Константа	Имя элемента	Константа	Имя элемента
16000010	bootdebug	25000080	safeboot	260000A1	halbreakpoint
16000017	noumex	250000C1	driverloadfailurepolicy	260000A2	useplatformclock
16000020	bootems	250000E0	bootstatuspolicy	260000B0	ems
16000040	advancedoptions	250000F0	hypervisorlaunchtype	260000F2	hypervisordebug
16000041	optionsedit	250000F3	hypervisordebugtype	260000F8	hypervisordisableslat
16000046	graphicsmodedisabled	250000F4	hypervisordebugport	27000030	customactions
16000048	nointegritychecks	250000F5	hypervisorbaudrate	31000003	ramdisksdidevice
16000049	testsigning	250000F6	hypervisorchannel	32000004	ramdisksdipath
16000050	extendedinput	25000100	tpmbootentropy	35000001	ramdiskimageoffset
1700000A	badmemorylist	25000120	xsavepolicy	35000002	ramdisktftpclientport
21000001	osdevice	25000121	xsaveaddfeature0	35000005	ramdiskimagelength
21000001	filedevice	25000122	xsaveaddfeature1	35000007	ramdisktftpblocksize
21000005	associatedosdevice	25000123	xsaveaddfeature2	35000008	ramdisktftpwindowsize
21000022	bcddevice	25000124	xsaveaddfeature3	36000006	exportascd
22000001	bpbstring	25000125	xsaveaddfeature4	36000009	ramdiskmcenabled
22000002	systemroot	25000126	xsaveaddfeature5	3600000A	ramdiskmctftpfallback
22000002	filepath	25000127	xsaveaddfeature6		
22000002	applicationname	25000128	xsaveaddfeature7		
		25000129	xsaveremovefeature		
		2500012A	xsaveprocessorsmask		
		2500012B	xsavedisable		

Доступ до елемента можна здійснювати через інтерфейс інструментарію управління Windows (WMI). Для перегляду вмісту сховища конфігурації можна скористатися командою:

bcdedit /enum all – відобразити всі записи в BCD

bcdedit /enum all > C:\enum-all.txt – те ж, що і у попередньому випадку, але з виведенням результатів у текстовий файл enum-all.txt на диску C:\.

Системний розділ – це розділ диска, на якому зберігаються файли, необхідні для запуску Windows (наприклад, Ntldr, Boot.ini і Ntdetect.com) (рис. 12.3).

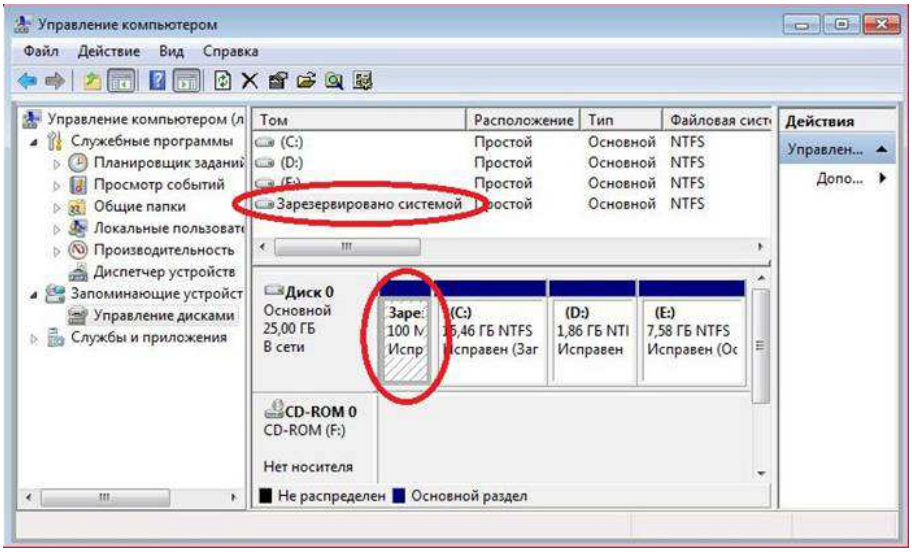


Рисунок 12.3 – Системний розділ

Обов’язкові умови створення прихованого системного розділу Windows:

1. Перша і основна умова – завантаження із зовнішнього пристрою (DVD, USB), оскільки під час запуску програми установки з-під Windows у вас не буде можливості працювати з розділами жорсткого диска.

2. Загальна кількість основних (первинних) розділів жорсткого диска перед початком установки не повинна перевищувати 3-х. Тобто, якщо до початку установлення простір вашого жорсткого диска вже поділений на 4 таких розділи, в цьому випадку прихований розділ розміром 100 Мб сформований не буде, а файли завантаження знаходитимуться на вже наявному активному розділі. Причому це може бути не той розділ, на який встановлюється система. Кількість логічних розділів на розширеному диску значення не має.

3. Розділ, на який виконується установка, має бути першим (верхнім, на графічному поданні дискового простору).

4. Установлення ОС повинно виконуватися на нерозмічену ділянку диска. Якщо диск вже розмічений, то при виборі розділу для установлення системи вам

необхідно не просто його відформатувати, але і створити наново, тобто розділ для установки потрібно спочатку видалити. Відповідно, якщо ви не хочете, щоб на жорсткому диску створювався розділ **System Reserved**, не видаляйте той, що є.

До програм сторонніх виробників, розгляду яких присвячена лабораторна робота, належать:

1. Victoria.
2. BOOTICE (by *www.ipauly .com*).
3. Windows Performance Toolkit.
4. Notmyfault.
5. Windows Debugger.

Victoria. Універсальна програма для тестування, діагностики і сервісного обслуговування IDE і Serial ATA вінчестерів. Орієнтована на широкий круг користувачів комп'ютерів. Працює з накопичувачем на низькому рівні (через порти контроллера) (рис. 12.4).

Основні можливості:

– автодетект PCI ATA/SATA контроллерів за кодом класу і підтримка 60-ти популярних моделей;

– виведення повної технічної інформації про жорсткий диск;

– десяток тестів для перевірки поверхні і «механіки» диска;

– створення і запис образу диска;

– перевірка пам'яті та інтерфейсу HDD;

– бенчмарк-функції;

– дефектоскоп поверхні;

– низькорівневе форматування HDD;

– виявлення і утаєння дефектів методом перепризначення секторів з резерву (remap);

– посекторное копіювання довільної області HDD у файл, з пропуском дефектних ділянок (корисно для порятунку інформації з пошкодженого диска);

– управління акустичним шумом;

– управління парольним захистом;

– зручний SMART-монітор;

– можливість зміни обсягу HDD;

– проглядання інформації про логічні розділи через порти;

– вбудований файловий менеджер;

– вбудована довідкова система.

Рисунок 12.4 – Головне вікно ***Victoria***

BOOTICE. Програма дозволить змінювати, створювати резервні копії і відновлювати пошкоджений головний завантажувальний запис Master Boot Record (MBR) і розділ Boot Record для локальних дисків або USB флеш-дисків.

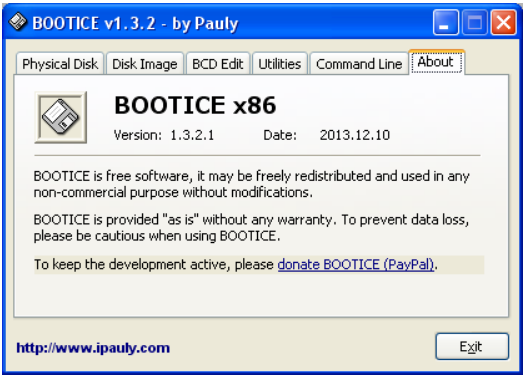


Рисунок 12.5 – Програма ***BOOTICE***

BOOTICE допоможе в розмічуванні і форматуванні USB жорстких дисків і флеш-карт, якщо раніше вони вже відформатували з файловою системою, яку не розпізнає операційна система Windows, що зазвичай призводить до того, що диск стає не видний в системі або видно не всі розділи. BOOTICE підтримує завантажувальні записи Grub4Dos, SysLinux, Plop, Windows NT5/6 та інші.

За допомогою функції Sector Editor в BOOTICE стає можливим відкриття редактора довільного сектора (рис. 12.6).

MBR, наприклад, для фізичного диска 0 (HD0: ST3500320AS) виглядатиме як зазначено на рис. 12.7.

Для проведення аналізу розділів MBR жорсткого диска дозволить відповідне меню. У загальному випадку структура головного завантажувального запису MBR зазначена у таблиці 12.1.

Таблиця 12.1 – Структура головного завантажувального запису MBR

Програма і дані початкового завантажувача (код завантажувача)	Таблиця розділів диска	55AA

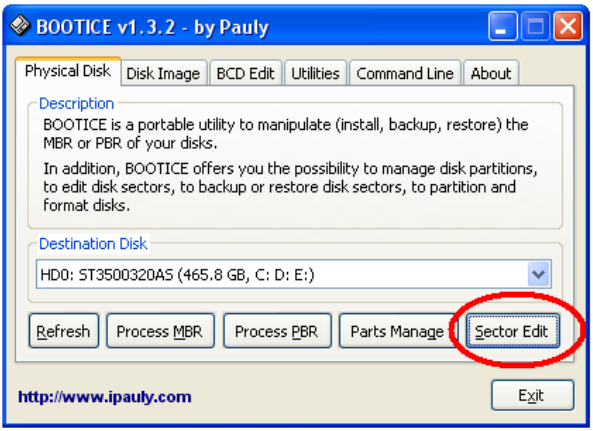


Рисунок 12.6 – Функція Sector Editor

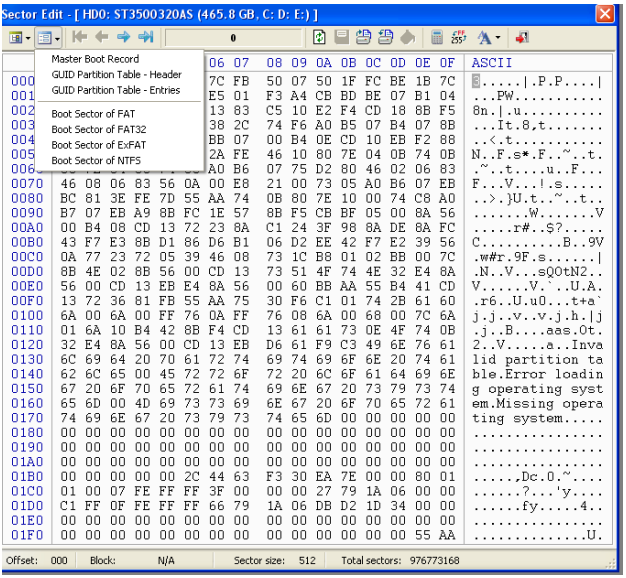


Рисунок 12.7 – MBR для фізичного диска

У випадку, якщо користувач не знає, що робить і до чого це призведе, то використовується BOOTICE тільки зі змінними, не системними жорсткими дисками.

Структура головного завантажувального запису MBR (табл. 12.2):

– програмний код і дані початкового завантажувача (446 байт);

– таблиця розділів диска (4 поля по 16 байт – 64 байти);

– сигнатура 55AA (2 байти).

Таблиця 12.2 – Структура головного завантажувального запису MBR

Зсув	Довжина, байт	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 і передати йому управління.

У таблиці розділів зберігається інформація про тип розділу і його розташування на жорсткому диску. Останні два байти MBR називаються ***сигнатурою***. Значення цих байтів має бути 55h AAh. У випадку, якщо це не так, запис прочитався некоректним. Структура опису розділу наведена в таблиці 12.3.

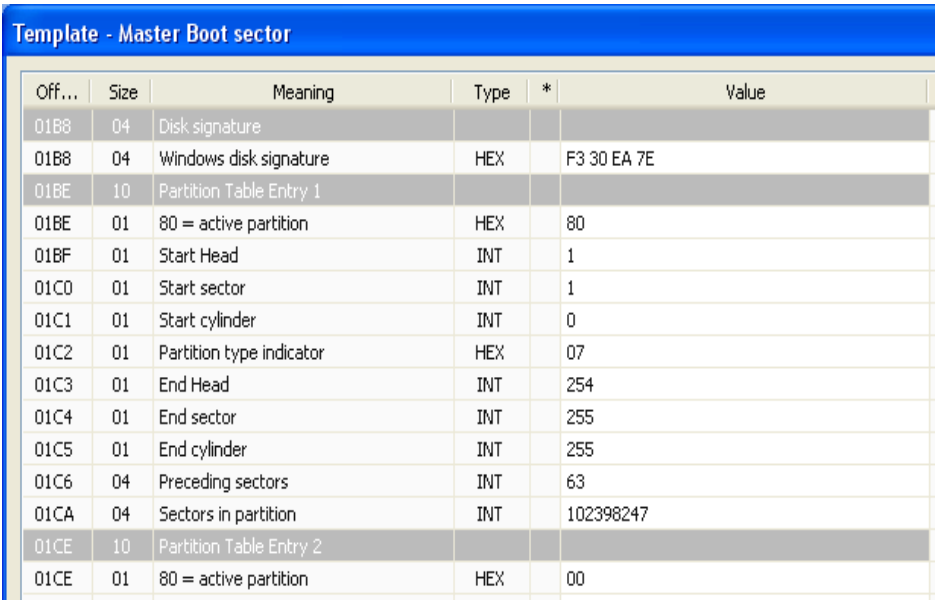
Таблиця 12.3 – Структура опису розділу

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, 7)
03h	1	Початок розділу – циліндр (старші біти 8, 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, 7)
07h	1	Кінець розділу – циліндр (старші біти 8, 9 зберігаються в байті номера сектора)
08h	4	Зсув першого сектора (Preceding sectors)
0Ch	4	Кількість секторів розділу (Section in partition)

Ознака активності – ознака, що позначає можливість завантаження опера-

ційної системи з даного розділу (рис. 12.8). Для стандартних завантажувачів набувє наступних значень:

- 80h – розділ є активним (active partition);
- 00h – розділ є неактивним;
- інші значення є помилковими і ігноруються.



Template - Master Boot sector

Off...	Size	Meaning	Type	*	Value
01B8	04	Disk signature			
01B8	04	Windows disk signature	HEX		F3 30 EA 7E
01BE	10	Partition Table Entry 1			
01BE	01	80 = active partition	HEX		80
01BF	01	Start Head	INT		1
01C0	01	Start sector	INT		1
01C1	01	Start cylinder	INT		0
01C2	01	Partition type indicator	HEX		07
01C3	01	End Head	INT		254
01C4	01	End sector	INT		255
01C5	01	End cylinder	INT		255
01C6	04	Preceding sectors	INT		63
01CA	04	Sectors in partition	INT		102398247
01CE	10	Partition Table Entry 2			
01CE	01	80 = active partition	HEX		00

Рисунок 12.8 – Майстер завантажувального сектора

Координати початку та кінця розділу подані в CHS-форматі (циліндр, головка, сектор). CHS не дозволяє виконувати адресацію більш ніж до 7,8 ГБ даних, і для адресації до розділів, що знаходяться за межами 7,8 ГБ, використовується LBA-адресація.

Код типу розділу, що визначає код файлової системи, використовуваної в даному розділі, надано у таблиці 12.4.

Таблиця 12.4 – Код типу розділу

Код	Тип розділу
00h	Порожній запис (вільне місце)
01h	FAT– 12
02h	XENIX root
03h	XENIX usr
04h	FAT– 16 до 32 Мбайт
05h	Розширений розділ
06h	FAT– 16 понад 32 Мбайт
07h	Windows NT NTFS (і деякі інші – тип визначається за вмістом завантажувального запису)
08h	AIX
09h	AIX завантажувальний
0Ah	OS/2 Boot менеджер
0Bh	FAT– 32
0Ch	FAT– 32 з використанням LBA
0Eh	FAT– 16 з використанням LBA (VFAT)
0Fh	Розширений розділ LBA (те ж що і 05h, з використанням LBA)

Windows Performance Toolkit – це програмне забезпечення, яке дозволяє значно прискорити роботу персонального комп'ютера. Windows за допомогою цієї утиліти працює швидше і стабільніше. Можливе відновлення втрачених даних і повний контроль над жорсткими дисками. Згладжує перебої роботи системи і усуває її недоліки. Стирає історію відвідувань сайтів, а так само очищає жорсткий диск від непотрібних файлів.

Windows Performance Toolkit розповсюджується у складі Windows SDK. Після установлення SDK з каталогу BIN необхідно встановити версію Windows Performance Toolkit відповідну для платформи та wpt_x86.msi, wpt_x64.msi або wpt_ia64.msi відповідно.

В основі Windows Performance Toolkit використано такі утиліти:

– **XPerf**, яка служить для активації збору інформації;

– **XBootMgr**, що дозволяє збирати інформацію в процесі завантаження комп'ютера, його виключення, переходу в режим Standby і виходу з режиму;

– **XPerfView**, використовується для візуального аналізу інформації продуктивності.

Взаємодія XPerf і XPerfView з системними компонентами – насамперед з підсистемою **Event Tracing for Windows** (ETW) – показана на рис. 12.9.

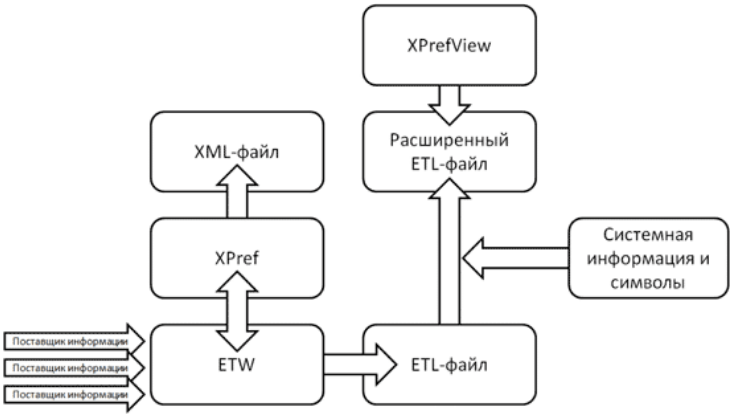


Рисунок 12.9 – Взаємодія утиліт XPerf і XPerfView з системними компонентами

Робота з утилітами XPerf і XPerfView відбувається таким чином:

1. За допомогою XPerf включаємо протоколювання подій на рівні ETW.

2. Виконуємо операції, що цікавлять нас, процеси і т.п..

3. За допомогою утиліти XPerf завершуємо сесію протоколювання.

4. Виконуємо обробку інформації – додавання до ETL-файлу, що створюється підсистемою ETW, системної інформації і символів.

5. Переглядаємо і аналізуємо результат за допомогою утиліти XPerfView.

Аналогічним чином працює і 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, і т.д.

ETL-файл, наприклад розміщується в теці C:\Trace. Тоді процес збору даних про завантаження системи запускається однією командою:

```
xbootmgr -trace boot -traceFlags BASE+CSWITCH+DRIVERS+POWER -resultPath C:\Trace
```

Аналогічні команди можна використовувати для діагностики **гібернації**:

```
xbootmgr -trace hibernate -traceFlags BASE+CSWITCH+DRIVERS+POWER -resultPath C:\Trace
```

та **сну**:

```
xbootmgr -trace standby -traceFlags BASE+CSWITCH+DRIVERS+POWER -resultPath C:\Trace
```

Комп'ютер буде перезавантажений. Якщо після входу в систему буде запит UAC від xbootmgr, то необхідно дозволити утиліті продовжити роботу. Через дві хвилини відобразиться вікно, показане на рис. 12.10.

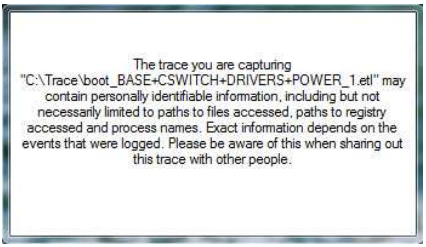


Рисунок 12.10 – Результат роботи утиліти

У теці C:\Trace мають бути три файли, як показано на рис. 12.11.

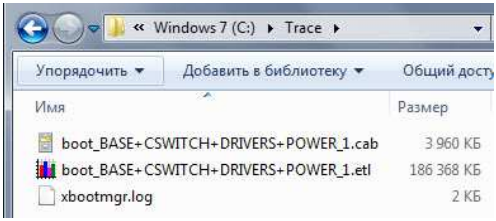


Рисунок 12.11 – Файли після закриття вікна утиліти

Для аналізу ETL-файлу використовується Performance Analyzer (утиліта **XPerfView**).

Графік **Boot Phases** відображає тривалість основних етапів завантаження. З графіка видно, що останній етап, **Post Boot**, зайняв 24,7 секунд (Duration), а загальний час завантаження склав майже 80 секунд (End Time) (рис. 12.12).

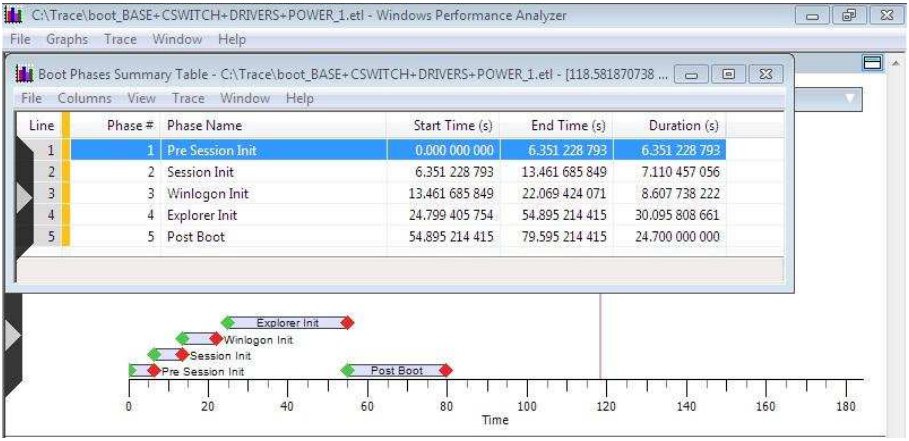


Рисунок 12.12 – Графік **Boot Phases**

На рис. 12.13 подані основні етапи завантаження, головний з них складається з чотирьох фаз.

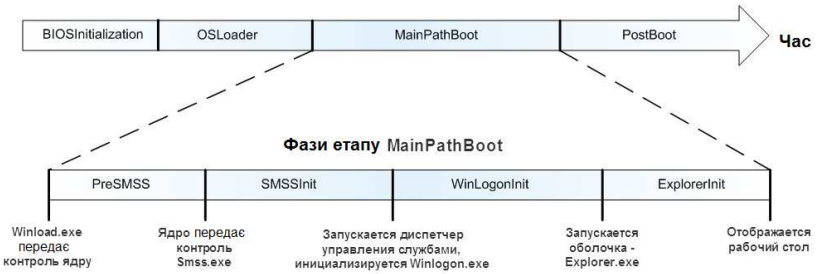


Рисунок 12.13 – Основні етапи завантаження ОС

Етап **OSLoader** йде відразу після ініціалізації BIOS. Візуально він починається після заставки і діагностичних екранів BIOS, а закінчується приблизно з появою екрана «Завантаження Windows».

На етапі **OSLoader**:

– завантажувач Windows (**winload.exe**) завантажує основні системні драйвери, які необхідні для зчитування мінімально необхідного набору даних з диска;

– завантажувач ініціалізував систему до моменту, з якого стає можливим завантаження ядра;

– ядро починає завантажуватися, **winload.exe** поміщає в оперативну пам'ять системний розділ реєстру і додаткові драйвери, помічені як BOOT_START.

Візуально етап **MainPathBoot** починається з екрана «Загрузка Windows» і завершується при появі робочого столу. Якщо не налаштовано автоматичний вхід в систему, тривалість цього етапу збільшується за рахунок часу, який потрібний для введення пароля.

Під час етапу MainPathBoot відбувається основна робота з завантаження операційної системи: ініціалізувалося ядро; відбувається визначення пристроїв Plug and Play (PNP); запускаються служби; виконується вхід в систему; ініціалізувався Explorer, тобто система готується до завантаження робочого столу.

Етап складається з чотирьох фаз, кожна з яких має власні характеристики і може по-своєму впливати на тривалість завантаження системи.

Фаза **PRESMSS** (у графічному зображенні WPT вона позначена як **Pre Session Init**) починається з ініціалізації ядра. Під час ініціалізації:

– ядро ініціалізує структури даних і компоненти й запускає диспетчер PNP;

– диспетчер PNP у свою чергу ініціалізує драйвери BOOT_START, які були завантажені 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 і екраном входу в систему, поя-

ва якого сигналізує про завершення фази.

Фаза **SMSSInit** (у графічному зображенні WPT вона позначена як **Session Init**) починається з того, що ядро передає контроль диспетчерові сесій (**smss.exe**). Під час цієї фази система:

– ініціалізує реєстр;

– завантажує і запускає пристрої і другу хвилю драйверів, які не помічені як BOOT_START;

– запускає процеси підсистеми.

Фаза завершується з передачею контролю процесу **winlogon.exe**.

«Синій екран смерті» (Blue Screen of Depth) – синій екран, що виникає при припиненні роботи Windows через катастрофічнийо збій або внутрішню помилку, що перешкоджає роботі системи.

Часто такою причиною стає посилання на адресу в пам'яті, що порушує права доступу, наприклад, спроба запису на сторінку, що має атрибут «тільки для читання», або спроба читання за ще не відображеною на пам'ять адресою. Іншою поширеною причиною є непередбачені виключення або переривання. Збої також трапляються, коли одна з підсистем ядра (така, як диспетчер пам'яті або диспетчер електроживлення) або драйвер (наприклад, USB-драйвер або драйвер дисплея) виявляють неузгодженість виконуваних ними операцій.

З якої б причини не відбувся збій 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 з 7 по 7 SP1, двадцять найчастіших стоп-кодів, що зустрічаються і є причиною 91 % системних відмов, можна розбити на такі категорії:

1. Звернення до відсутньої сторінки. Стоп-коди в цьому випадку:

- 0xA – IRQL_NOT_LESS_OR_EQUAL;
- 0xD1 – DRIVER_IRQL_NOT_LESS_OR_EQUAL.

2. Управління електроживленням Стоп-код:

0x9F – DRIVER_POWER_STATE_FAILURE.

3. Виключення і системні переривання. Стоп-коди:

- 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 з параметром P1, не рівним 0xC0000005 STATUS_ACCESS_VIOLATION.

4. Порушення прав доступу. Стоп-коди:

- 0x50 – PAGE_FAULT_IN_NONPAGED_AREA;
- 0x8E – KERNEL_MODE_EXCEPTION_NOT_HANDLED з параметром P1, рівним 0xC0000005 STATUS_ACCESS_VIOLATION.

5. Дисплей. Стоп-код: 0x116 – VIDEO_TDR_FAILURE.

6. Пул. Стоп-коди:

- 0x19 – BAD_POOL_HEADER;
- 0xC2 – BAD_POOL_CALLER;
- 0xC5 – DRIVER_CORRUPTED_EXPOOL.

7. Управління пам’яттю. Стоп-коди:

- 0x1A – MEMORY_MANAGEMENT;
- 0x4E – PFN_LIST_CORRUPT.

8. Апаратне забезпечення.

- 0x7A – KERNEL_DATA_INPAGE_ERROR;
- 0x124 – WHEA_UNCORRECTABLE_ERROR.

9. USB. Стоп-код: 0xFE – BUGCODE_USB_DRIVER.

10. Важливий об’єкт. Стоп-код:

0xF4 – CRITICAL_OBJECT_TERMINATION.

11. Файлова система NTFS. Стоп-код: 0x24 – NTFS_FILE_SYSTEM.

За умовчанням усі системи сімейства Windows налаштовані на запис інфор-

мації про стан на момент збою.

При завантаженні система отримує параметри аварійного дампу з розділу HKLM\SYSTEM\CurrentControlSet\Control\CrashControl реєстру. Якщо генерація дампа задана, створюється копія драйвера міні-порту диска, через яку том записується в пам'ять. Копія отримує ім'я міні-порту з приставкою «dump».

Різні збої можна викликати за допомогою програми Notmyfault (рис. 12.14), створеної у Windows Sysinternals. Вона складається з виконуваного файлу Notmyfault.exe і драйвера Myfault.sys. Після запуску цей файл завантажує драйвер і виводить показане на рисунку діалогове вікно, що дозволяє різними способами припинити роботу системи або викликати її зависання, а також змусити драйвер ініціювати втрату пам'яті з вивантажуваного або невивантажуваного пулу.

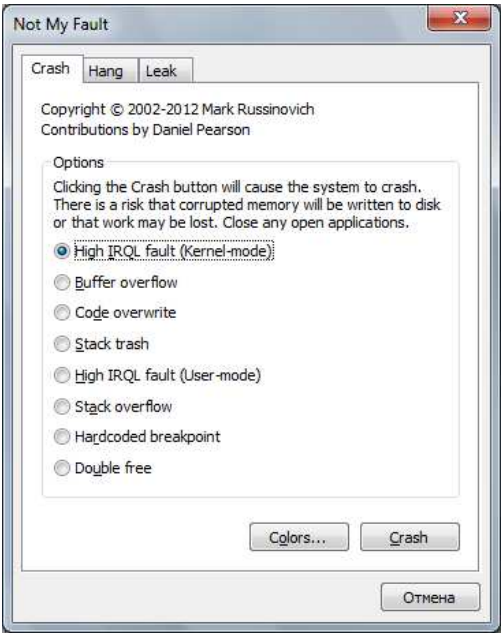


Рисунок 12.14 – Програма Notmyfault

Найпростіший збій в додатку Notmyfault викликається установленням перемикача High IRQL Fault (Kernel-Mode) і клацанням на кнопці Crash (рис. 12.15).

Після цього драйвер виділяє сторінку в пулі підкачуваної пам'яті, збільшує IRQL-рівень до DPC/dispatch і звертається до звільненої сторінки. Якщо це не приводить до збою, процес продовжує прочитувати пам'ять після кінця сторінки, поки не відбудеться збій через звернення до недійсної сторінки. В результаті драйвер виконує декілька недопустимих операцій:

1. Посилається на пам'ять, яка йому не належить.

2. Звертається до пулу підкачуваної пам'яті на IRQL-рівні DPC/dispatch і вище, що недопустимо – на цих рівнях помилки сторінок не вирішуються.

3. Виходить за межі виділеної ділянки пам'яті і намагається звернутися до пам'яті, яка потенційно може бути недійсною.

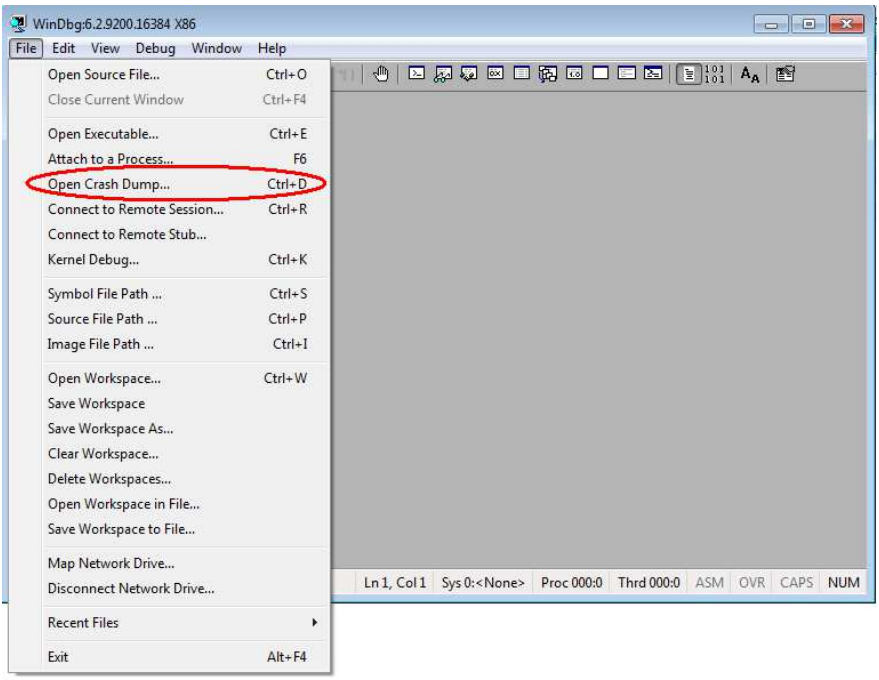


Рисунок 12.15 – Відкриття аварійного дампа

Перше звернення до сторінки не завжди призводить до збою, тому що звільнена драйвером сторінка може залишитися в системному робочому наборі.

Після завантаження, що згенерував цим збоєм аварійного дампа, у налагоджувач WinDbg буде виведено 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 operation
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. Following frames may be wrong.

92926b3c 904dc5ab badb0d00 8412d240 93013008 nt+0x467eb

92926bb8 904dc9db 85d9a4f0 92926bfc 904dcb26 myfault+0x5ab

92926bc4 904dcb26 85d844a8 00000001 00000000 myfault+0x9db

92926bfc 8288c4bc 853289b0 85d9a4f0 85d9a4f0 myfault+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 налагоджувач як для додатків, так і драйверів. Безліч плагінів, команд, скриптова мова. WinDbg можна використовувати як дебаггер додатків призначеного для користувача режиму. Сам дебаггер містить величезну кількість команд, але використовується дуже мала їх частина. Докладніший опис можливо виявити в рекомендованій літературі.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

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 (переповнення 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 корзини)

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 точуи зупинки)

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 (переповнення 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 корзини)

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 (переповнення стека)

ВАРІАНТ 15

1. {hypervisorsettings}

2. Ім’я фази: Post Boot

3. Причина BSOD: hardcoded breakpoint (помилка захисту точки зупинки).

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 (переповнення 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 (переповнення стека)

ВАРІАНТ 23

1. {emssettings}
2. Ім’я фази: Winlogon Init
3. Причина BSOD: hardcoded breakpoint (помилка захисту точки зупинки)

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 (переповнення 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 корзини)

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 (переповнення стека)

ХІД РОБОТИ

1. Аналітична частина

Етап 1. Провести аналіз головного завантажувального запису (MBR) жорсткого диска за допомогою програми Victoria. Результати, підтверджені скриншотами, занести до таблиці 12.5.

Таблиця 12.5 – Характеристики жорсткого диска

Standard					Advanced (Partition viewer)						
Model	Firmware	Serial	Disk size	CHS	N	Boot	System	Start LBA	End LBA	Size	Name
					1						
					2						
					..						

За допомогою програми BootIce прочитавши MBR, подати скриншот таблиці розділів жорсткого диска. Зробити висновок про ідентичність (або відмінності) в значеннях однотипних параметрів різних програм.

Етап 2. Провести аналіз головного завантажувального запису (MBR) Flash накопичувача за допомогою програми Victoria. Провести тестування свого Flash накопичувача. Результати, підтверджені скриншотами, занести до таблиці 12.6.

Таблиця 12.6 – Характеристики Flash накопичувача

Standard						
Model	Firmware	Serial	Disk size	CHS	Features	Sector

2. Дослідницька частина

Етап 3. Провести дослідження елементів системного сховища конфігураційних даних (BCD store) для вказаного в **п.1** індивідуального завдання об’єкта

BCD. Результати занести до таблиці 12.7.

Таблиця 12.7 – BCD-параметри для об'єкту {*}

GUID об'єкту – { - - - - }			
Ім'я елемента	Константа	Значення	Призначення
device	11000001	partition=Z:	пристрій завантаження
.....	.....	.....	.....

* **п.1** індивідуального варіанта завдання.

Етап 4. Провести дослідження фаз завантаження операційної системи. Результати, підтверджені скраншотами, занести до таблиці 12.8.

Таблиця 12.8 – 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. Провести визначення переліку завантажуваних драйверів, використовуючи меню Driver Delays, на етапі (фазі), який задано в **п.2** індивідуального завдання, час виконання яких не перевищує 400 ms. Результати, підтверджені скриншотами, занести до таблиці 12.9. На скриншотах відобразити ТІЛЬКИ ті, що задовольняють дану умову драйвера.

Таблиця 12.9 – Завантажувані драйвера під час фази _____

№ фази	Ім'я драйвера	Тривалість, с

1	Ntfs.sys	307
2		

Етап 6. Провести базовий аналіз аварійного дампа, отриманого внаслідок збою, що штучно згенерував синій екран смерті. Використовувати програму Notmyfault з драйвером myfault.sys, а причину відмови задати відповідно до **п.3** індивідуального варіанта завдання. Подати скриншот BSOD, лістинг аварійного дампа (без автоматичних коментарів).

Контрольні запитання

1. Поясніть відмінність етапів завантаження систем на базі BIOS і UEFI.
2. Назвіть основні компоненти процесу завантаження 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 в завантаженні відіграють процеси smss.exe, csrss.exe, wininit.exe?
6. Яким чином у системному реєстрі задається автоматичний запуск програм за умовчанням?
7. Як здійснюється протоколювання завантаження в безпечному режимі?
8. У якому файлі відображається і фіксується стан завантаження?
9. Чому у Windows трапляються збої?
10. Які види файлів аварійного дампа?

СПИСОК ЛІТЕРАТУРИ

Основна

1. Таненбаум Э. Современные операционные системы / Э. Таненбаум – СПб.: Питер, 2010. – 1120 с.

2. Руссинович М. Внутреннее устройство Microsoft Windows. Ч.1 / М. Руссинович, Д. Соломон : пер. с англ. – СПб. : Питер, 2013. – 800 с.

3. Руссинович М. Внутреннее устройство Microsoft Windows. Ч.2. Основные подсистемы ОС / М. Руссинович, Д. Соломон, А. Ионеску : пер. с англ. – СПб. : Питер, 2014. – 672 с.

4. Шеховцов В. А. Операційні системи / В. А. Шеховцов – Київ : Видавнича група BHV, 2005. – 576 с.

5. Побегайло А. П. Системное программирование в Windows / А. П. Побегайло – СПб. : БХВ-Петербург, 2006. – 1056 с.

6. Олифер В. Г. Сетевые операционные системы / В. Г. Олифер, Н. А. Олифер – СПб. : Питер, 2006. – 544 с.

Додаткова

7. Маклин Й. Установка и настройка Windows 7. Учебный курс Microsoft / Й. Маклин, Т. Орин. – М. : Русская редакция, 2011. – 848 с.

8. Руссинович М. Внутренне устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. Мастер класс / М. Руссинович, Д. Соломон : пер. с англ. – М.: Изд.-торг. дом «Русская Редакция», 2005. – 992 с.

9. Саймон Р. Windows 2003 API. Энциклопедия программиста / Р. Саймон : пер. с англ. – Киев : ООО «ДиасофтЮП», 2004. – 1088 с.

10. Побегайло А.П. Системное программирование в Windows / А. П. Побегайло – СПб.: БХВ-Петербург, 2006. – 1056 с.

11. Бэкон Дж. Операционные системы / Дж. Бэкон, Т. Харрис – Киев : Издат.группа BHV; СПб. : Питер, 2004. – 800 с.

12. Кокорева О.И. Реестр Windows XP / О. И. Кокорева – СПб.: БХВ-Петербург, 2004. – 560 с.

13. Олифер В. Г. Сетевые операционные системы / В. Г. Олифер, Н. А. Олифер – СПб. : Питер, 2002. – 544 с.

14. Столингс В. Операционные системы / В. Столингс – М. : Вильямс, 2002. – 848 с.

15. Голубничий Д.Ю. Системне програмування і операційні системи. Ч.1.: навч. пос. / Д.Ю. Голубничий, В.Ф. Третяк. – Харків: Вид. ХДЕУ, 2004. – 192 с.

16. Голубничий Д.Ю. Системне програмування та операційні системи. Ч.2.: навч. пос. / Д.Ю. Голубничий, В.Ф. Третяк, С.В. Кавун. – Харків: Вид. ХНЕУ, 2005. – 264 с.

17. Сорокина С.И. Программирование драйверов и систем безопасности: учеб. пособ. /С. И. Сорокина, А. Ю. Тихонов, А. Ю. Щербаков. – СПб.: БХВ-Петербург, 2003. – 256 с.

18. Попов А.В. Введение в Windows PowerShell / А. В. Попов – СПб.: БХВ-Петербург, 2009. – 464 с.

19. Джонсон М. Разработка приложений в среде Linux / М. Джонсон, Э. Троян : пер. с англ. – М. : ООО «И.Д. Вильямс», 2007. – 544 с.

20. Секунов Н.Ю. Программирование на C++ в Linux / Н. Ю. Секунов – СПб.: БХВ-Петербург, 2004. – 368 с.

Навчальне видання

ГОЛУБНИЧИЙ Дмитро Юрійович
ХОЛОДКОВА Анна Валеріївна
ШМАТКО Олександр Віталійович
КОЗУЛЯ Марія Михайлівна

ОПЕРАЦІЙНІ СИСТЕМИ

Лабораторний практикум
для студентів спеціальності 122 «Комп'ютерні науки»
та 121 «Програмна інженерія»
усіх форм навчання

Відповідальний за випуск проф. *М. Д. Годлевський*
Роботу до видання рекомендував проф. *О. В. Горілий*
Редактор *О. С. Самініна*

План 2019 р., поз. 20

Підп. до друку 05.07.2019 р. Формат 60×84 1/16. Папір офсетний.
Riso-друк. Гарнітура Times New Roman. Ум. друк. арк. 9,95 .
Наклад 100 прим. Зам. № 194426. Ціна договірна.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2

Видавництво «Стильна типографія»
61002, м. Харків, вул. Чернишевська, 28А. Тел.: (057) 754-49-42
e-mail: zebraprint.zakaz@gmail.com
Свідоцтво суб'єкта видавничої справи: серія ДК №5493 від 22.08.2017 р.