МІНІСТЕРСТВО ОСВІТИ І НАУКИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

МЕТОДИЧНІ ВКАЗІВКИ
ЩОДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«СИСТЕМНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ»
ДЛЯ СТУДЕНТІВ ДЕННОЇ ТА ЗАОЧНОЇ ФОРМ НАВЧАННЯ
ЗІ СПЕЦІАЛЬНОСТІ 123 – «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»
ЧАСТИНА ІІІ

КРЕМЕНЧУК 2020

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Системне програмне забезпечення» для студентів денної та заочної форм навчання зі спеціальності 123 – «Комп'ютерна інженерія». Частина ІІІ.

Укладачі: старш. викл. Ю. В. Зілінський,
асист. Р. С. Цвєнтарний
Рецензент к.т.н., доц. Ю. В. Лашко

Кафедра комп’ютерних та інформаційних систем

Затверджено методичною радою КрНУ імені Михайла Остроградського
Протокол № _____ від «___ » _____ 20__ р.
Голова методичної ради _____ проф. В. В. Костін

ЗМІСТ

ВСТУП

Виконання лабораторних робіт з курсу «Системне програмне забезпечення» сприяє поглибленню теоретичних знань, які студенти отримують у лекційному матеріалі та при самостійному читанні спеціальної літератури, і значною мірою дозволяє закріпити матеріал практичних занять. Крім цього, студенти одержують практичні навички розробки й налагодження програм, які необхідні при виконанні курсового проекту з даного курсу та лабораторних робіт з інших дисциплін.

Для виконання лабораторних робіт потрібне знання програмної моделі, системи команд і способів адресації мікропроцесорів з архітектурою IA32, а при роботі з довідковою системою засобів розробки також знадобиться знання синтаксису і базових конструкцій мови програмування C/C++.

Основу засобів розробки, які використовуються в лабораторному практикумі, складають пакети асемблерів, побудовані на основі Microsoft Macro Assembler зі складу Windows Driver Kits, налагоджувач OllyDbg та засоби налагодження зі складу Debugging Tools for Windows або Windows Driver Kits, інтеґроване середовище розробки програм RadAsm IDE.

Заздалегідь, згідно з розкладом занять, перед виконанням уже безпосередньо експериментальної частини лабораторної роботи студенти повинні підготуватися до її виконання. Підготовка до лабораторної роботи проводиться студентом самостійно у позааудиторний час. Підготовка складається з вивчення відповідних розділів лекційного матеріалу, коротких теоретичних відомостей, що наведені як безпосередньо у даних методичних вказівках, так і в літературі, зазначеній у відповідному розділі кожної лабораторної роботи. У ході підготовки до лабораторної роботи студент також повинен розпочати складання звіту.

Допуск студента до виконання експериментальної частини лабораторної роботи здійснюється викладачем. Студент повинен подати звіт з підготовки до виконання лабораторної роботи, який містить назву й мету роботи та відповіді

на контрольні питання. У протилежному випадку студент уважається не готовим до лабораторної роботи, до виконання її експериментальної частини не допускається і продовжує підготовку безпосередньо у лабораторії під керівництвом викладача.

За результатами виконання експериментальної частини студент складає звіт, зміст якого зазначений у відповідному розділі кожної лабораторної роботи. Залік з лабораторної роботи студент одержує після співбесіди з викладачем, у ході якої він повинен продемонструвати теоретичні знання з теми даної лабораторної роботи, обґрунтувати вибір алгоритму розв’язання задачі, дати вичерпні пояснення за змістом звіту з виконання лабораторної роботи.

Тематика лабораторних робіт даних методичних вказівок взаємопов’язана з тематикою практичних занять з курсу та охоплює питання перехоплення функцій програмного інтерфейсу та програмування сервісів ОС Windows NT.

Третя частина є логічним продовженням другої частини цих методичних вказівок і охоплює останні сім лабораторних робіт, передбачених для виконання у шостому навчальному семестрі робочою навчальною програмою дисципліни «Системне програмне забезпечення» для бакалаврів зі спеціальності 123 – «Комп'ютерна інженерія» (у тому числі скорочений термін навчання).

1 ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

Лабораторна робота № 1

Тема. Перехоплення API-функцій шляхом модифікації таблиці імпорту додатка

Мета: набуття практичних навичок перехоплення API-функцій за допомогою заміни оригінальних адрес функцій у таблиці імпорту додатка.

Короткі теоретичні відомості

Одним з поширених методів перехоплення функцій, окрім розглянутого у попередніх лабораторних роботах методу модифікації машинного коду функції (splicing), є метод модифікації адрес функцій у таблиці імпорту додатка.

Нагадаємо, що програмний інтерфейс додаткам (API) ОС Windows NT надає через клієнтські DLL підсистем оточення, використовуючи для цього два механізми: раннє (статичне) і пізнє (динамічне) зв'язування. Виклик API функцій при статичній компоновці з DLL (раннє зв'язування) виконується наступним чином. Оскільки на етапі створення додатка лінкеру не відома адреса функції, що викликається, він резервує місце для цієї адреси в так званій таблиці імпорту (IAT), а всі виклики функції у додатку спрямовує на створений запис в IAT. Адреса функції засобами ОС визначається та заноситься до таблиці імпорту на етапі підготовки додатка до виконання. Схема виклику API-функції зображена на рисунку 1.1.

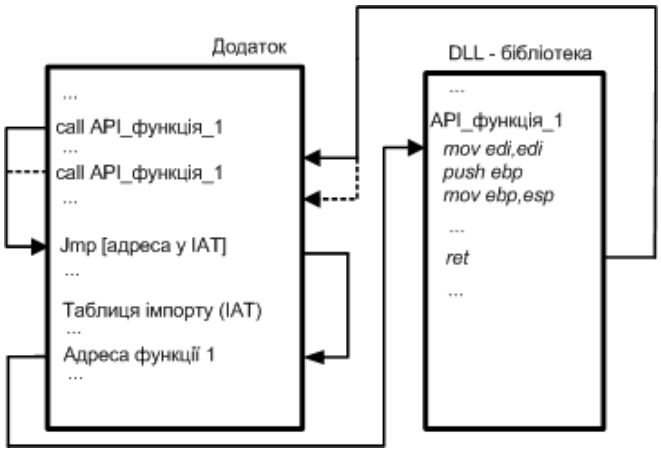


Рисунок 1.1 – Схема виклику API-функції

Перехоплення шляхом модифікації точок входу функцій в IAT базується на заміні адреси оригінальної функції на адресу функції перехоплювача. У такому випадку під час виклику функції керування без відома додатка передається перехоплювачеві, якому надається можливість повного контролю результату виконання оригінальної функції до повернення його додатку. Схему такого перехоплення зображено на рисунку 1.2.

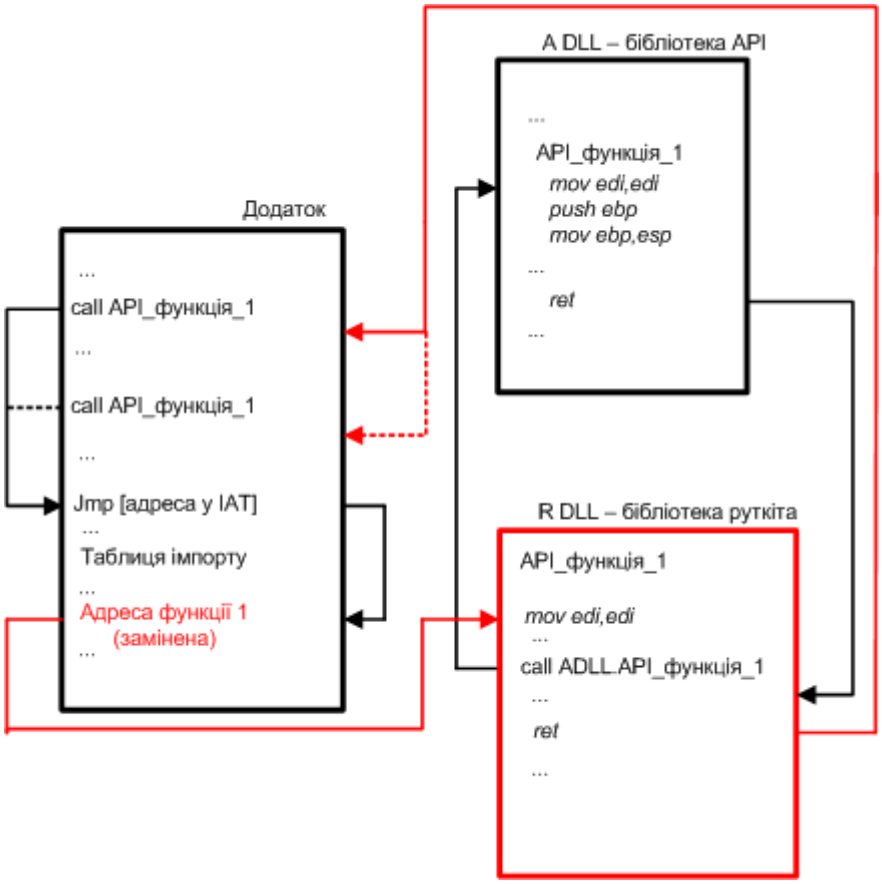


Рисунок 1.2 – Схема перехоплення API-функції шляхом модифікації IAT

Для реалізації вищеописаного перехоплення необхідно знати структуру таблиці імпорту додатка. Таблиця імпорту є складовою частиною PE-заголовка програмного файла. Якщо розробникові відома структура PE-заголовка, яка є досить складною, то знайти таблицю імпорту можна безпосередньо в ньому. Але існує більш зручний спосіб пошуку таблиці імпорту, оснований на використанні виклику API-функції ImageDirectoryEntryToData, яка експортується бібліотекою dbghelp.dll або imagehlp.dll.

PVOID ImageDirectoryEntryToData(

IN LPVOID Base,

IN BOOLEAN MappedAsImage,

IN USHORT DirectoryEntry,

OUT PULONG Size)

Base – базова адреса модуля, інформацію про який необхідно отримати. Цю адресу можна отримати за допомогою виклику GetModuleHandle.

MappedAsImage – може приймати два значення: TRUE або FALSE. Якщо значення цього параметра дорівнює TRUE, то модуль завантажено (відображено на пам’ять) завантажувачем ОС, якщо FALSE, то відображення виконано викликом MapViewOfFile.

DirectoryEntry – константа, яка визначає, яку саме інформацію необхідно отримати з PE-заголовка. Для отримання інформації про таблицю імпорту необхідно передати значення IMAGE_DIRECTORY_ENTRY_IMPORT.

Size – змінна, у яку повертається розмір у байтах блока інформації, що запитується.

У разі невдачі функція повертає NULL. У разі успішного виклику функція повертає покажчик на масив структур IMAGE_IMPORT_DESCRIPTOR, що несуть інформацію про бібліотеки, з яких імпортуються функції модуля.

```
IMAGE_IMPORT_DESCRIPTOR STRUCT
    union
        Characteristics         dd  ?
        OriginalFirstThunk      dd  ?
    ends
        TimeDateStamp   dd  ?
    ForwarderChain  dd  ?
    Name1           dd  ?
    FirstThunk      dd  ?
IMAGE_IMPORT_DESCRIPTOR ENDS
```

Поле Name1 – це покажчик відносно базової адреси завантаження додатка (параметр Base у виклику ImageDirectoryEntryToData) на ім’я DLL, яка експортує для нього функції.

Інформація про наступну бібліотеку міститься у наступній структурі IMAGE_IMPORT_DESCRIPTOR, яка розташовується слідом за першою, і т. д. Ознакою закінчення масиву структур є нульове значення у полі Name1.

На рисунку 1.3 зображено схему пошуку у таблиці імпорту додатка точки входу у функцію з іменем FuncName, яка імпортується з бібліотеки DLLName2.dll. Кожна стрілка на схемі є покажчиком.

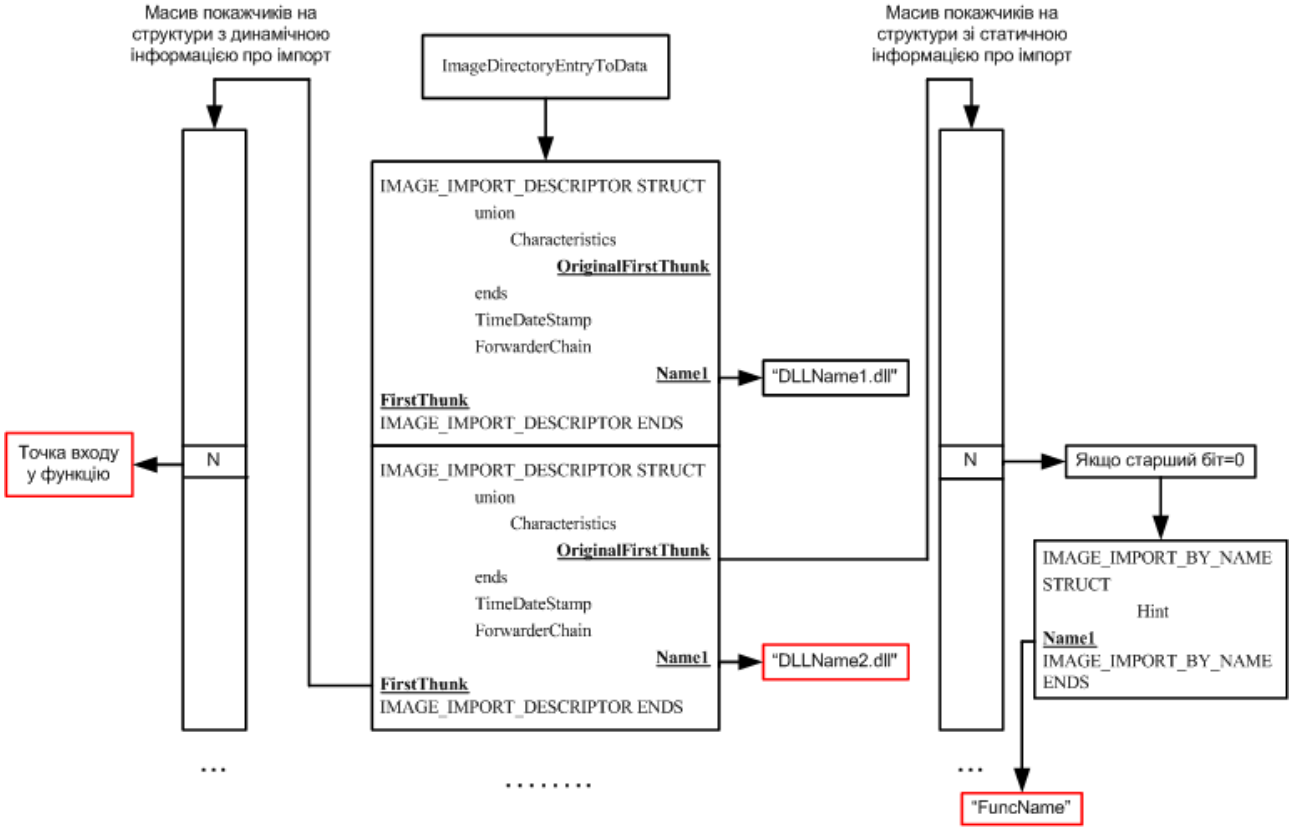


Рисунок 1.3 – Схема пошуку точки входу у функцію в таблиці імпорту

Для перехоплення певної функції необхідно знати, яка бібліотека її експортує, і виконати пошук структури IMAGE_IMPORT_DESCRIPTOR, поле Name1 якої вказує на рядок з ім’ям бібліотеки.

Коли знайдена потрібна структура IMAGE_IMPORT_DESCRIPTOR, необхідно використати її поле OriginalFirstThunk, яке є покажчиком відносно базової адреси додатка на масив покажчиків на структури

IMAGE_THUNK_DATA32, які несуть статичну інформацію про функції, що імпортуються, – про ім’я або ординал.

```
IMAGE_THUNK_DATA32 STRUCT
union u1
    ForwarderString  dd  ?
    Function         dd  ?
    Ordinal          dd  ?
    AddressOfData    dd  ?
ends
IMAGE_THUNK_DATA32 ENDS
```

Паралельно з масивом покажчиків на структури IMAGE_THUNK_DATA32 зі статичною інформацією про функції, що імпортуються, існує масив покажчиків на такі ж самі структури IMAGE_THUNK_DATA32, які несуть динамічну інформацію про функції, що імпортуються, а саме – про адреси точок входу у функції. Адреса (відносно базової адреси модуля) початку масиву покажчиків на структури з цією динамічною інформацією міститься у полі FirstThunk структури IMAGE_IMPORT_DESCRIPTOR.

Для пошуку за ім’ям функції, яку потрібно перехопити, необхідно скористатися статичною інформацією про імпорт. Для цього треба аналізувати подвійні слова, на які вказують елементи масиву покажчиків на структури IMAGE_THUNK_DATA32. Якщо це подвійне слово дорівнює 0, то це ознака закінчення масиву. Якщо старший біт подвійного слова, на яке вказує покажчик, дорівнює 1, то імпорт ведеться за ординалом, а не за ім’ям. Якщо ж цей біт дорівнює 0, то це подвійне слово є покажчиком відносно базової адреси додатка на структуру IMAGE_IMPORT_BY_NAME.

```
IMAGE_IMPORT_BY_NAME STRUCT
    Hint        dw  ?
    Name1       dd  ?
IMAGE_IMPORT_BY_NAME ENDS
```

Тут Name1 – адреса рядка, що містить ім’я функції.

Якщо ім’я функції, на яке вказує поле Name1 структури IMAGE_IMPORT_BY_NAME, не збігається з іменем функції, яку треба перехопити, то необхідно продовжити пошук, тобто перейти до наступного елемента у масиві покажчиків на структури IMAGE_THUNK_DATA32 зі статичною інформацією про імпорт. При цьому зручно паралельно просуватися і на наступний елемент у масиві покажчиків на структури IMAGE_THUNK_DATA32, які несуть динамічну інформацію про імпорт.

Коли знайдено необхідний елемент масиву структур IMAGE_THUNK_DATA32 зі статичною інформацією про імпорт, то елемент з тим самим індексом у паралельному масиві структур IMAGE_THUNK_DATA32 з динамічною інформацією про імпорт містить покажчик на шукану точку входу у функцію. Цю точку входу і необхідно замінити адресою функції перехоплювача.

Порядок виконання роботи

1. Завантажити із сервера кафедри архів з файлами поточної лабораторної роботи і розкрити його у будь-який каталог свого облікового запису.
2. Відкрити каталог проекту ChangeIATApp.
3. Відкрити файл проекту ChangeIATApp.rap і ознайомитися з його реалізацією.
4. Запустити командний інтерпретатор cmd.exe та перейти у каталог проекту ChangeIATApp.
5. Запустити Блокнот за допомогою командного інтерпретатора, набравши у його вікні команду notepad.exe.
6. Виконати у консолі командного інтерпретатора команду inject.exe notepad.exe ChangeIATApp.dll.
7. У Блокноті набрати будь-який текст і закрити його вікно, попередньо не зберігаючи файл, і занести до звіту результат виконання цього пункту.

8. Узявши за основу проект InjectDll попередньої частини цих методичних вказівок, розробити додаток RemoveDll, який буде виконувати дії, протилежні до додатка InjectDll, тобто відвантажувати вказану DLL з адресного простору вказаного додатка шляхом використання віддаленого потоку.
9. Повторити пункти 5, 6 та після цього за допомогою розробленого у попередньому пункті додатка відвантажити бібліотеку ChangeIATApp.dll з адресного простору Блокноту. Повторити пункт 7 та занести до звіту пояснення щодо результатів його виконання.
10. Модернізувати проект ChangeIATApp таким чином, щоб бібліотека коректно відвантажувалася з адресного простору додатка.
11. Перевірити правильність відвантаження бібліотеки ChangeIATApp.dll з адресного простору додатка за допомогою розробленого додатка RemoveDll.

Зміст звіту

Зміст із цієї лабораторної роботи повинен містити:

1. Назву і мету лабораторної роботи.
2. Відповіді на контрольні питання.
3. Результати (за необхідності у вигляді Screenshot), отримані при виконанні лабораторної роботи, та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проекту додатка і проекту модернізованої DLL-бібліотеки з їх детальними коментарями.

Контрольні питання

1. Що таке PE-заголовок програмного файла?
2. Що таке таблиця імпорту і навіщо вона використовується?
3. Яким чином ОС виконує виклик API-функції при статичному (ранньому) зв’язуванні з бібліотекою, яка експортує функцію?
4. У якому випадку перехоплення функцій за допомогою модифікації таблиці імпорту буде безрезультатним?

5. Чи є таблиця імпорту обов’язковою для кожного додатка Windows NT?
6. Чи можна знайти таблицю імпорту без використання виклику ImageDirectoryEntryToData?
7. Що таке і для чого використовується DIT (Delay Import Table)?
8. У чому переваги та недоліки перехоплення функцій методом модифікації таблиці імпорту порівняно з методом модифікації машинного коду функцій (сплайсингу)?

Література: [1, с. 36–50; 4, с. 506–518, с. 564–580; 5, с. 160–169].

Лабораторна робота № 2

Тема. Перехоплення API-функцій шляхом модифікації таблиці імпорту модулів, які експортують функції для додатка

Мета: набуття практичних навичок перехоплення API-функцій за допомогою заміни оригінальних адрес функцій у таблиці імпорту модулів, які експортують функції для додатка.

Короткі теоретичні відомості

Перехоплення API-функцій шляхом модифікації таблиці імпорту додатка не завжди дозволяє досягти поставленої мети. Наприклад, з попередніх лабораторних робіт відомо, що додаток «Блокнот» використовує функцію SetClipboardData для роботи з буфером обміну Windows, і її перехоплення методом сплайсінгу коду дає можливість контролю цього буфера. Функція SetClipboardData експортується системною бібліотекою user32.dll, але незважаючи на те, що таблиця імпорту додатка «Блокнот» містить цілу низку функцій цієї бібліотеки, серед них немає функції SetClipboardData, і виконати перехоплення цієї функції шляхом модифікації таблиці імпорту додатка «Блокнот» неможливо.

З цього стає очевидним, що виклик функції у додатку може виконуватися не напряму, а, наприклад, з деякої іншої функції, яка, можливо, експортується іншою бібліотекою, а вже ця бібліотека, у свою чергу, імпортує необхідну функцію.

Тому для забезпечення більш надійного перехоплення необхідно вміти модифікувати не тільки таблицю імпорту додатка, а й таблиці імпорту модулів (бібліотек), які використовує додаток (рисунок 2.1).

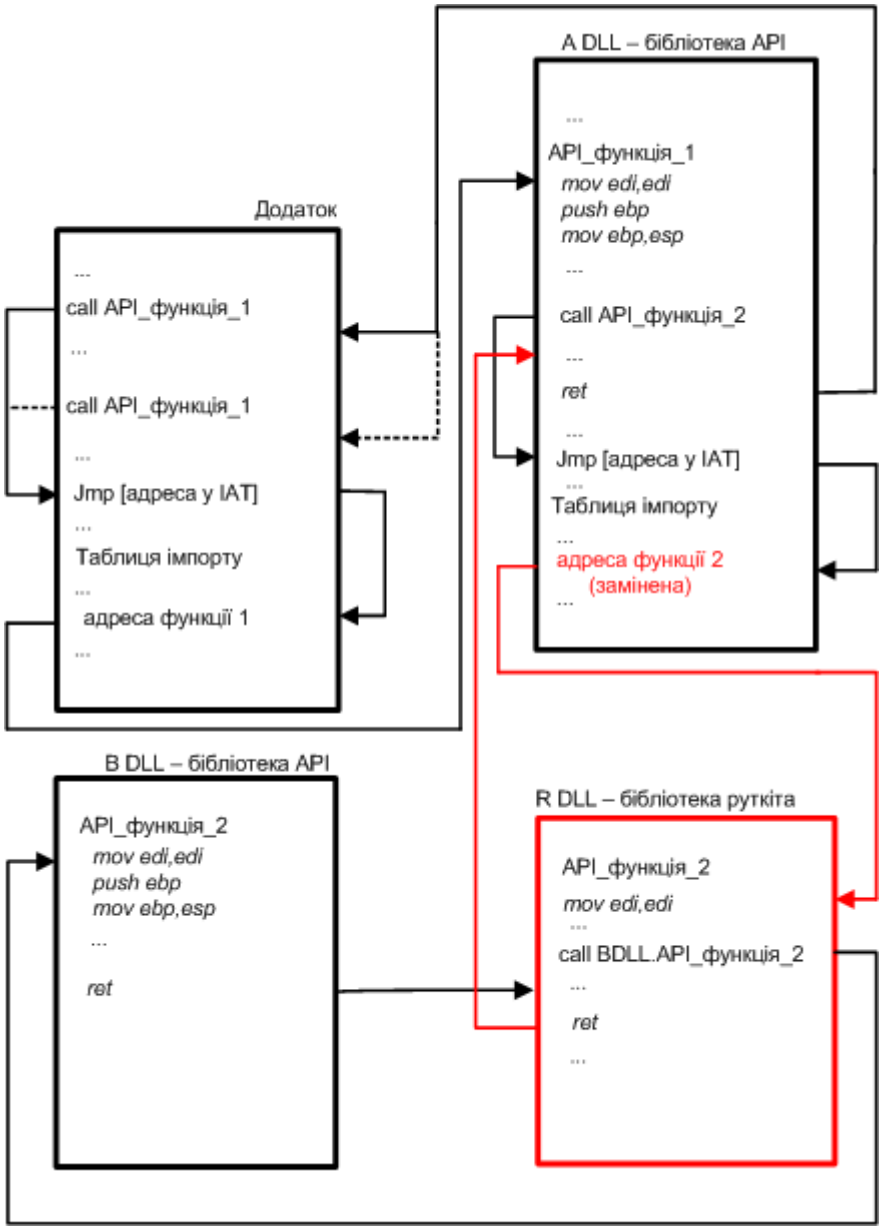


Рисунок 2.1 – Схема перехоплення API-функції шляхом модифікації таблиці імпорту модуля додатка

Таблиця імпорту модуля додатка за структурою нічим не відрізняється від таблиці імпорту самого додатка. Тому її модифікація виконується тим самим способом, який описано у попередній лабораторній роботі.

Порядок виконання роботи

1. Завантажити із сервера кафедри архів з файлами поточної лабораторної роботи і розкрити його у будь-який каталог свого облікового запису.

2. Відкрити каталог проекту ChangeIATDLL.
3. Відкрити файл проекту ChangeIATDLL.rap і ознайомитися з його реалізацією.
4. З меню «Пуск» запустити додатки «Блокнот» і «Калькулятор» та переглянути списки їх модулів, використовуючи для цього додаток procexp.exe. Занести до звіту результати виконання цього пункту.
5. За допомогою додатка inject.exe виконати впровадження бібліотеки ChangeIATDLL.dll у процес explorer.exe.
6. За допомогою додатка procexp.exe переглянути список модулів процесу explorer.exe та переконатися, що бібліотека ChangeIATDLL.dll завантажена в адресний простір explorer.exe.
7. Скопіювати бібліотеку ChangeIATDLL.dll у каталог користувача (echo %USERPROFILE% у консолі cmd.exe) та повторити пункт 4. Додатково занести до звіту пояснення щодо необхідності зазначеного копіювання та відмінностей у списках модулів порівняно з першим виконанням пункту 4.
8. За допомогою додатка RemoveDLL.exe, розробленого у попередній лабораторній роботі, відвантажити бібліотеку ChangeIATDLL.dll з адресного простору explorer.exe.
9. Спробувати запустити будь-який додаток з меню «Пуск» та занести до звіту пояснення щодо результатів виконання цього пункту.
10. Модернізувати проект ChangeIATDLL таким чином, щоб бібліотека коректно відвантажувалася з адресного простору додатка.
11. Перевірити правильність відвантаження бібліотеки ChangeIATDLL.dll з адресного простору додатка.

Зміст звіту

Зміст із цієї лабораторної роботи повинен містити:

1. Назву і мету лабораторної роботи.
2. Відповіді на контрольні питання.

3. Результати (за необхідності у вигляді Screenshot), отримані при виконанні лабораторної роботи, та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проекту модернізованої DLL-бібліотеки з їх детальними коментарями.

Контрольні питання

1. Які переваги дає перехоплення функцій у таблицях імпорту системних бібліотек?
2. Нехай у додатку А модифікована таблиця імпорту системної бібліотеки kernel32.dll і перехоплено деякі функції. Додаток Б при цьому використовує ті ж самі функції бібліотеки kernel32.dll. Яким чином буде вести себе додаток Б під час виклику цих функцій, якщо додаток А продовжує роботу і не повернув таблицю імпорту бібліотеки kernel32.dll у початковий стан?
3. Чи можна розглянутим у цій лабораторній роботі методом перехоплювати функції у модулі ntdll.dll?
4. Яким чином можна уникнути аварій у додатку при відвантаженні бібліотеки-перехоплювача?
5. Яким чином можна уникнути аварій у додатку при виникненні помилок у роботі функції-перехоплювача, не пов’язаних з помилками програмування?
6. У яких каталогах та у якому порядку виконується пошук DLL при зв’язуванні?

Література: [1, с. 50–54; 4, с. 564–580].

Лабораторна робота № 3

Тема. Використання стандартних хуків

Мета: набуття практичних навичок використання стандартних хуків Windows на прикладі перехоплення віконних повідомлень клавіатури.

Короткі теоретичні відомості

У попередніх лабораторних роботах були розглянуті методи перехоплення API-функцій операційних систем Windows NT, які основані на особливостях виклику функцій програмного інтерфейсу операційної системи і є більшою мірою наслідком цих особливостей, а не передбаченим механізмом для виконання перехоплення.

Однак у сімействі ОС Windows NT є і документована можливість стеження за роботою інших додатків, яка основана на використанні так званих хуків (hook). Хуки є стандартним методом перехоплення віконних повідомлень графічних додатків. Оскільки перехоплювати можна лише віконні повідомлення, то стає очевидним, що можливості хуків з точки зору контролю сторонніх додатків обмежені порівняно з розглянутими раніше методами перехоплення функцій програмного інтерфейсу API.

Процедури хуків, які встановлюються для перехоплення віконних повідомлень сторонніх додатків, реалізуються у DLL. Для встановлення хука у ланцюжок хуків використовується функція:

```
HHOOK SetWindowsHookEx(
int idHook,
HOOKPROC lpfn,
HINSTANCE hMod,
DWORD dwThreadId)
```

idHook – визначає тип хука, що встановлюється. Це значення задається одною із констант WH_* і визначає, які саме віконні повідомлення потрібно перехоплювати.

lpfn – покажчик на процедуру хука.

hMod – хендл DLL, у якій знаходиться процедура хука.

dwThreadId – ідентифікатор потоку, за віконними повідомленнями якого стежить хук. Якщо необхідно стежити за віконними повідомленнями всіх потоків у системі, це значення має бути нульовим.

У разі успішного виклику функція повертає хендл установленого хука, який використовується в інших функціях для роботи із хуками. У разі невдалого виклику функція повертає NULL.

Процедура хука, адреса якої міститься у lpfn, має однаковий прототип, незалежно від типу віконного повідомлення, що перехоплюється:

```
LRESULT CALLBACK ProcName(
    int code,
    WPARAM wParam,
    LPARAM lParam)
```

Проте параметри процедури code, wParam, lParam для кожного віконного повідомлення несуть різний зміст.

Наприклад, для перехоплення віконних повідомлень клавіатури тип хука має буди заданий константою WH_KEYBOARD. При цьому параметр code може приймати значення HC_ACTION або HC_NOREMOVE. В обох випадках wParam містить віртуальний код натиснутої клавіші, а lParam містить повну інформацію про натиснуту клавішу (кількість повторень, скан-код, чи є клавіша натиснутою або відпущеною, попередній стан клавіші та ін.). Якщо значення параметра code є HC_NOREMOVE, то це означає, що віконне повідомлення від клавіатури не було вилучене із системної черги повідомлень, тобто додаток відправив це повідомлення за допомогою функції PostMessage зі встановленим прапорцем PM_NOREMOVE.

Незалежно від типу віконного повідомлення, що перехоплюється, слід пам’ятати про те, що *при code<1 процедура хука не повинна обробляти повідомлення* і має передати його наступному хуку в ланцюзі хуків за допомогою виклику функції CallNextHookEx та повернути результат, який повертає ця функція.

```
LRESULT CallNextHookEx(
    HHOOK hhk,
    int nCode,
    WPARAM wParam,
```

LPARAM lParam)

hhk – хендл поточного встановленого хука, а інші параметри збігаються з параметрами процедури поточного хука.

Значення, яке необхідно повернути із процедури хука, залежить від типу хука. Цю інформацію можна знайти у довідковій системі за функціями Win API.

Якщо для хука типу WH_KEYBOARD процедура хука повертає нульове значення, то Windows повертає віконне повідомлення цільовій процедурі обробки віконних повідомлень. Якщо ж з процедури хука повертається ненульове значення, то це блокує передачу повідомлення наступному хуку в ланцюзі хуків або цільовій процедурі обробки віконних повідомлень.

Для видалення хука з ланцюга хуків використовується функція BOOL UnhookWindowsHookEx(HHOOK hhk), де hhk – хендл раніше встановленого хука. У разі успішного виклику повертається ненульове значення, у разі невдалого – нульове.

Порядок виконання роботи

1. Завантажити із сервера кафедри архів з файлами поточної лабораторної роботи і розкрити його у будь-який каталог свого облікового запису.
2. Запустити додаток KeyRegistration.exe.
3. Запустити додатки «Блокнот» та «Microsoft Word», набрати у їх вікнах будь-який текст.
4. Переглянути вікно регістратора KeyRegistration.exe та занести до звіту пояснення щодо його вмісту.
5. Відкрити каталог проекту Hook та ознайомитися з реалізацією проекту Hook.rap.
6. Відкрити каталог проекту KeyRegistration та ознайомитися з реалізацією проекту KeyRegistration.rap.
7. Модернізувати проект KeyRegistration таким чином, щоб після заповнення буферної сторінки пам’яті виконувалося збереження накопичених даних у файл KeyRegistration.log і сторінка починала

заповнюватися з початку. При завершенні додатка KeyRegistration.exe вся інформація, що знаходиться у буферній сторінці пам’яті, також повинна записуватися до файла KeyRegistration.log. Крім того, модернізувати проект KeyRegistration таким чином, щоб замість хендлу вікна, з якого отримується потік клавіатурного введення, регіструвався текст заголовка цього вікна, якщо він є. Для цього необхідно скористатися функцією GetWindowText.

8. Модернізувати проекти KeyRegistration та Hook таким чином, щоб вони, замість уручну заданого номера повідомлення WM_USER+1, використовували для зв’язування між собою унікальний номер повідомлення, наданий операційною системою. Для цього необхідно скористатися функцією RegisterWindowMessage.

Зміст звіту

Зміст із цієї лабораторної роботи повинен містити:

1. Назву і мету лабораторної роботи.
2. Відповіді на контрольні питання.
3. Результати (за необхідності у вигляді Screenshot), отримані при виконанні лабораторної роботи, та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проектів модернізованого додатка та DLL з їх детальними коментарями.

Контрольні питання

1. У чому переваги та недоліки методу перехоплення за допомогою хуків порівняно з раніше розглянутими методами перехоплення?
2. У якому випадку і навіщо у процедурі хука необхідно викликати функцію CallNextHookEx?
3. Чи може процедура хука бути реалізованою у звичайному виконуваному exe-файлі? Відповідь обґрунтувати.

4. Які з перерахованих додатків ОС сімейства Windows NT і чому можна та не можна контролювати за допомогою хуків: calc.exe, cmd.exe, notepad.exe, ping.exe, tracert.exe, explorer.exe?
5. Чи може додаток:
    а) установити декілька різних хуків;
    б) установити декілька хуків, процедури яких знаходяться у різних DLL;
    в) установити хук на тип віконного повідомлення, на який уже встановлено хук іншим додатком?
6. Чи може:
    а) декілька процедур хуків знаходитися в одній DLL;
    б) декілька хуків перехоплювати один тип віконних повідомлень;
    в) один хук перехоплювати декілька типів віконних повідомлень?

Література: [1, с. 28–34, с. 105–112; 4, с. 537–549].

Лабораторна робота № 4

Тема. Інфраструктура сервісних процесів

Мета: знайомство з інфраструктурою, що забезпечує функціонування сервісів Windows NT; набуття практичних навичок програмної реалізації програм керування сервісами Windows NT.

Короткі теоретичні відомості

До інфраструктури, що забезпечує роботу сервісів Windows NT, входять наступні компоненти:

1. Власне сервісний процес – служба.
2. Реєстр.
3. Диспетчер керування сервісами.
4. Програма керування сервісом.
5. Системні сервісні процеси (служби).
6. Стандартні програми керування сервісами.

На кінцевому етапі завантаження системи, перед появою діалогу реєстрації користувача, запускається Диспетчер керування сервісами (Service Control Manager – SCM), який переглядає розділ реєстру сервісів HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services.

На основі даних цього розділу реєстру Диспетчер керування сервісами створює свою внутрішню базу служб (Services Active Database або SCM database), далі знаходить у створеній базі всі служби, позначені для автоматичного запуску, і запускає їх. Дані про системні служби заносяться до реєстру при установці ОС. Для установки інших служб потрібна їхня реєстрація у системному реєстрі.

Рисунок 4.1 демонструє реєстрацію у реєстрі системи служби, яка розглядається у даній лабораторній роботі.

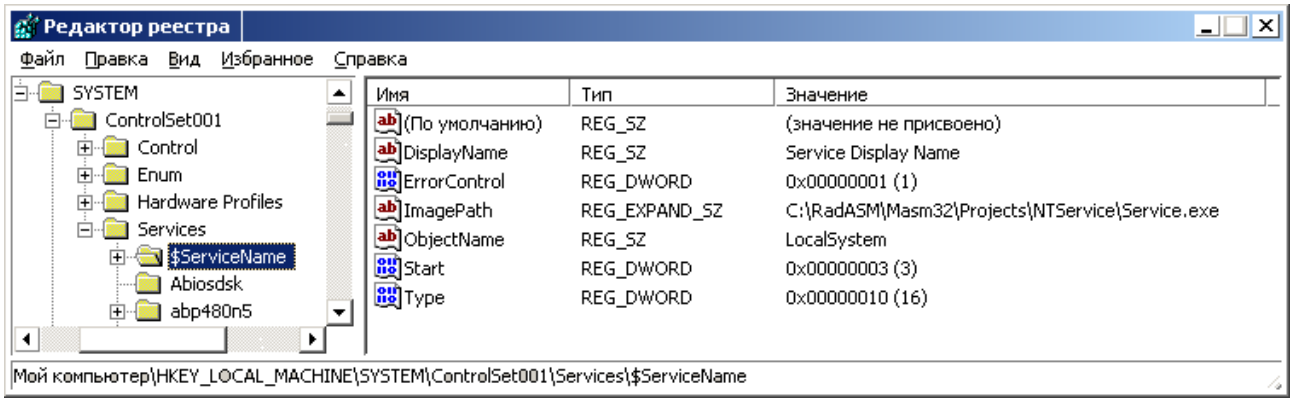


Рисунок 4.1 – Реєстрація сервісу в реєстрі системи

Параметр DisplayName визначає екранне ім'я сервісу, яке у нашому випадку дорівнює «Service Display Name». Екранне ім'я це довільна назва сервісу, яка відображується стандартними утилітами для роботи із сервісами.

Параметр ErrorControl визначає поводження системи при невдалій спробі запуску сервісу і може набувати наступних значень:

00000000h (SERVICE_ERROR_IGNORE) – ігнорувати помилку і продовжити роботу;

00000001h (SERVICE_ERROR_NORMAL) – повідомити користувачеві про невдалу спробу запуску сервісу, занести помилки до системного реєстраційного журналу і продовжити роботу;

00000002h (SERVICE_ERROR_SEVERE) – занести помилки до системного реєстраційного журналу й автоматично перевантажити систему, використовуючи останню вдалу конфігурацію. Якщо після перезавантаження сервіс не запускається, ігнорувати й продовжити завантаження далі;

00000003h (SERVICE_ERROR_CRITICAL) – занести помилки до системного реєстраційного журналу й автоматично перевантажити систему, використовуючи останню вдалу конфігурацію. Якщо після перезавантаження сервіс не запускається, перервати завантаження.

Параметр ImagePath визначає повне ім'я образу програмного файла процесу сервісу, яке у нашому випадку дорівнює C:\RadAsm\masm32\ Projects\NTService\Service.exe.

Параметр ObjectName визначає групу безпеки сервісу, яка у нашому випадку дорівнює «LocalSystem».

Параметр Start – визначає тип запуску сервісу і може набувати наступних значень:

00000000h і 00000001h стосується драйверів і для служб не використовується;

00000002h (SERVICE_AUTO_START) – сервіс повинен запускатися при завантаженні операційної системи;

00000003h (SERVICE_DEMAND_START) – сервіс буде запускатися вручну викликом функції StartService або автоматично при спробі почати роботу із залежними від нього сервісами;

00000004h (SERVICE_DISABLED) – запуск сервісу заборонений.

Параметр Type визначає тип сервісу і може набувати одного з наступних значень:

SERVICE_KERNEL_DRIVER (00000001h) – драйвер режиму ядра;

SERVICE_WIN32_OWN_PROCESS (00000010h) – в образі файла процесу служб реалізована одна служба;

SERVICE_WIN32_SHARE_PROCESS (00000020h) – в образі файла сервісного процесу реалізовано кілька служб;

SERVICE_INTERACTIVE_PROCESS (00000100h) – служби процесу можуть взаємодіяти з робочим столом користувача.

Також параметр Type може бути комбінацією всіх інших значень, окрім SERVICE_KERNEL_DRIVER.

Реєстрація сервісу у системі може бути виконана внесенням відповідних записів до реєстру за допомогою редактора реєстру Windows або подібних йому утиліт. Також можна виконати відповідну модифікацію системного реєстру програмно, використовуючи передбачені для цього в програмному інтерфейсі ОС засоби. Однак ні в першому, ні в другому випадку модифіковані дані не потраплять у внутрішню базу драйверів і служб SCM, і буде потрібно перезавантаження системи для того, щоб зміни набули чинності, що, наприклад, є недопустимим для працюючих безупинно серверів.

При створенні служби, як правило, розробляється й окремий додаток, за допомогою якого виконується установка служби й керування нею. Цей додаток називають програмою керування сервісом (Service Control Program – SCP). При установці служби або програмного забезпечення, що використовує службу, SCP, як мінімум, повинна виконати реєстрацію служби в системі.

При установці служби в системі SCP повинна:

а) установити зв'язок з SCM;

б) зареєструвати службу у базі даних SCM;

в) запустити службу.

За необхідності SCP також повинна передбачати можливість зупинки служби і її видалення із системи.

Для установки каналу зв'язку з SCM використовується функція OpenSCManager:

```
SC_HANDLE OpenSCManager(
        LPCTSTR lpszMachineName,
        LPCTSTR lpszDatabaseName,
        DWORD fdwDesiredAccess)
```

lpszMachineName – покажчик на рядок, що завершується нулем і містить ім’я комп'ютера. NULL, якщо канал зв'язку із SCM буде відкриватися тільки на локальному комп'ютері.

lpszDatabaseName – покажчик на рядок, що завершується нулем і містить ім’я бази даних. Може бути або SERVICES_ACTIVE_DATABASE, або NULL, що те ж саме, тому що в цьому випадку також буде відкрита база SERVICES_ACTIVE_DATABASE.

fdwDesiredAccess – запитувані права доступу:

SC_MANAGER_CONNECT – доступ на установку каналу зв'язку з SCM. Із цим рівнем доступу можна запускати сервіс, зупиняти і навіть видаляти відомості про нього з бази даних SCM і з реєстру. Цей вид доступу встановлюється завжди, навіть якщо він не зазначений явно.

SC_MANAGER_CREATE_SERVICE – доступ на занесення до бази даних SCM нового сервісу.

Для реалізації основних функцій SCP і цих двох прав доступу досить, а група Адміністратори взагалі має повний (SC_MANAGER_ALL_ACCESS) доступ до SCM.

Якщо канал зв’язку із SCM успішно встановлений, функція OpenSCManager повертає ідентифікатор, що надає доступ до активної бази даних SCM, в іншому випадку – NULL. Після використання ідентифікатора він повинен бути закритий за допомогою BOOL CloseServiceHandle(SC_HANDLE hSCObject).

Після одержання доступу до бази SCM реєстрація у ній нового сервісу виконується за допомогою функції

```
SC_HANDLE CreateService(
    SC_HANDLE       hSCManager,
    LPCTSTR         lpServiceName,
    LPCTSTR         lpDisplayName,
    DWORD           dwDesiredAccess,
    DWORD           dwServiceType,
```

DWORD dwStartType,
DWORD dwErrorControl,
LPCTSTR lpBinaryPathName,
LPCTSTR lpLoadOrderGroup,
LPDWORD lpdwTagId,
LPCTSTR lpDependencies,
LPCTSTR lpServiceStartName,
LPCTSTR lpPassword)

hSCManager – ідентифікатор бази даних SCM.

lpServiceName – покажчик на рядок, що завершується нулем і містить внутрішнє ім'я сервісу. Максимальна довжина рядка 256 символів. Не допускаються символи "/" й "\". Відповідає імені підрозділу в гілці HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services (див. рисунок 4.1).

lpDisplayName – покажчик на рядок, що завершується нулем і містить екранне ім'я. Максимальна довжина рядка 256 символів. Відповідає параметру DisplayName у реєстрі (див. рисунок 4.1).

dwDesiredAccess – тип доступу, який визначає права щодо маніпулювання створеним сервісом:

SERVICE_ALL_ACCESS – максимальний доступ;

SERVICE_START – доступ на запуск служби викликом функції StartService;

SERVICE_STOP – доступ на зупинку служби викликом функції ControlService з параметром SERVICE_CONTROL_STOP;

DELETE – доступ на видалення відомостей про службу з бази даних SCM викликом функції DeleteService.

dwServiceType – відповідає параметру Type у реєстрі.

dwStartType – відповідає параметру Start у реєстрі.

dwErrorControl – відповідає параметру ErrorControl у реєстрі.

lpBinaryPathName – відповідає параметру ImagePath у реєстрі.

lpLoadOrderGroup – покажчик на рядок, що завершується нулем і містить ім'я групи, членом якої є сервіс. Групи визначають порядок завантаження сервісів, які до них входять. Щоб запустити службу після завантаження системних драйверів і служб, її не потрібно включати ні до якої групи, указавши NULL або покажчик на порожній рядок.

lpdwTagId – для служби в цьому параметрі потрібно вказати NULL.

pDependencies – покажчик на масив розділених нулями імен сервісів або груп сервісів, які система повинна запустити до запуску цього сервісу. Цей масив повинен завершуватися двома нулями. Якщо сервіс не залежить від інших сервісів, то в цьому параметрі можна вказати NULL або покажчик на порожній рядок.

lpServiceStartName – покажчик на рядок, що завершується нулем і містить ім'я облікового запису, із правами якого повинна запускатися служба. Для запуску служби з обліковим записом LocalSystem цей параметр установлюється рівним NULL.

lpPassword – покажчик на рядок, що завершується нулем і містить пароль до облікового запису, зазначеного в параметрі lpServiceStartName. Для LocalSystem повинен бути NULL.

Запуск служби виконується за допомогою функції

```
BOOL StartService(
    SC_HANDLE     schService,
    DWORD         dwNumServiceArgs,
    LPCTSTR       *lpszServiceArgs)
```

schService – ідентифікатор сервісу. Цей ідентифікатор повертається CreateService, або його можна одержати за допомогою функції OpenService.

dwNumServiceArgs – кількість рядків-аргументів у масиві lpszServiceArgs. Якщо lpszServiceArgs дорівнює NULL, то й цей параметр також нульовий.

lpszServiceArgs – покажчик на масив, що містить покажчики на закриті нулем рядки, які будуть передані службі як аргументи.

Керування службою виконується за допомогою функції

```
BOOL ControlService(
    SC_HANDLE          hService,
    DWORD              dwControl,
    LPSERVICE_STATUS   lpServiceStatus);
```

hService – ідентифікатор служби, що повертається функціями CreateService або OpenService.

dwControl – один з керуючих кодів (вибірково):

SERVICE_CONTROL_STOP – зупинка сервісу;

SERVICE_CONTROL_PAUSE – припинення сервісу;

SERVICE_CONTROL_CONTINUE – поновлення після припинення;

SERVICE_CONTROL_INTERROGATE – запит стану сервісу;

SERVICE_CONTROL_SHUTDOWN – оповіщення про необхідність завершення роботи сервісу протягом 20 секунд внаслідок завершення роботи операційної системи;

SERVICE_CONTROL_PARAMCHANGE – оповіщення про зміну параметрів сервісу.

lpServiceStatus – адреса структури SERVICE_STATUS, у яку повертається інформація про поточний стан сервісу.

Видалення сервісу виконується за допомогою функції BOOL DeleteService(SC_HANDLE hService), де hService – ідентифікатор служби, що повертається функціями CreateService або OpenService.

Порядок виконання роботи

1. Завантажити із сервера кафедри архів з файлами поточної лабораторної роботи й розкрити його у будь-який каталог свого облікового запису.
2. Виконати запуск редактора реєстру regedit.exe і з його допомогою знайти у реєстрі та занести до звіту всі гілки та параметри зі словом $ServiceName.
3. Виконати запуск додатка StartService.exe з каталогу NTService.
4. За допомогою редактора реєстру знайти у реєстрі, переглянути і занести до звіту параметри гілки реєстру

HKLM\SYSTEM\CurrentControlSet\Services\$ServiceName.

5. Відкрити файл проекту StartService.rap каталогу NTService\StartService та ознайомитися з вмістом усіх файлів проекту. Зазначити у звіті функції додатка StartService.exe.
6. Закрити regedit.exe і відкрити консоль MMC для керування службами C:\WINDOWS\system32\services.msc.
7. Знайти у консолі MMC службу з екранним іменем «Service Display Name». Передивитися її властивості та занести до звіту відомості про поточний стан служби, внутрішнє (не екранне) ім’я служби, тип запуску служби, тип облікового запису, від імені якого виконується служба, дії системи при виникненні збоїв у роботі служби і залежності від інших служб.
8. Виконати в консолі MMC призупинення, відновлення роботи, зупинку, перезапуск служби «Service Display Name». Занести до звіту результат виконання всіх дій з керування службою.
9. Виконати у консолі командного інтерпретатора наступну послідовність команд:

   net pause $ServiceName

   net continue $ServiceName

   net stop $ServiceName

   net start $ServiceName

   і занести до звіту протокол сеансу роботи командного інтерпретатора та призначення виконаних команд.
10. Зупинити службу «Service Display Name» і передивитися вміст файла C:\service.log.
11. Запустити службу «Service Display Name» і закрити консоль MMC.
12. Виконати запуск додатку ControlService.exe у каталозі NTService і зазначити у звіті функції додатку.
13. Виконати запуск редактора реєстру regedit.exe і з його допомогою переглянути та занести до звіту параметри гілки реєстру

HKLM\SYSTEM\CurrentControlSet\Services\$ServiceName.

14. Відкрити консоль MMC для керування службами і знайти службу з екранним іменем «Service Display Name». Занести до звіту результати пошуку.
15. Відкрити файл проекту ControlService.rap каталогу NTService\ControlService та ознайомитися з вмістом усіх файлів проекту. Переконатися, що при виконанні п. 12 завдання у звіті були зазначені всі функції додатка ControlService.exe.
16. Розробити консольний додаток CommandService для керування службою «Service Display Name». Додаток повинен передавати службі команди з кодами, які передаються йому в командному рядку. Синтаксис командного рядка: CommandService.exe code nnn, де code – ключове слово, що є ознакою коду команди, а nnn – номер команди.

Зміст звіту

Зміст із цієї лабораторної роботи повинен містити:

1. Назву і мету лабораторної роботи.
2. Відповіді на контрольні питання.
3. Результати (за необхідності у вигляді Screenshot), отримані при виконанні лабораторної роботи, та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проекту розробленого додатка з його детальними коментарями.

Контрольні питання

1. Призначення параметрів реєстру у гілці реєстрації служби.
2. Чому при встановленні служби будь-яким способом потрібно реєструватися у системі з правами облікового запису групи адміністраторів?
3. Як можна тимчасово заборонити завантаження служби при старті операційної системи?

4. У чому перевага установки служби за допомогою власного додатка-інсталятора порівняно з іншими методами?
5. Чи можна сумістити програму керування службою і службу в одному виконуваному файлі (відповідь пояснити)?
6. Чи можна у параметри lpServiceName та lpDisplayName функції CreateService передати один і той самий рядок?
7. Для керування службою утилітами sc.exe та net.exe використовують ім’я, яке:
   а) передається функції CreateService у параметрі lpServiceName;
   б) передається функції CreateService у параметрі lpDisplayName;
   в) передається функції RegisterServiceCtrlHandler у параметрі lpServiceName;
   г) збігається з назвою гілки служби у реєстрі у кущі HKLM\SYSTEM\CurrentControlSet\Services.

Література: [2, с. 607–630; 3, с. 62–132; 7, с. 430–450].

Лабораторна робота № 5

Тема. Локальні інтерфейси сервісних процесів

Мета: набуття практичних навичок програмної реалізації локальних інтерфейсів сервісних процесів Windows NT.

Короткі теоретичні відомості

Сервісні процеси розглядаються як засоби, що дозволяють спростити реалізацію серверних частин клієнт-серверних додатків, й Microsoft рекомендує реалізовувати всі серверні додатки у вигляді служб. У цьому випадку для взаємодії клієнтської частини з реалізованою у вигляді служби серверною частиною використовуються методи організації IPC, які були розглянуті в лабораторних роботах попередніх частин цих методичних вказівок.

Аналізуючи розглянуті раніше методи організації IPC, слід зазначити, що іменовані канали пропонують ряд можливостей, які дозволяють розглядати їх як універсальний механізм для таких задач. Більш того реалізація іменованих

каналів у Windows NT узагалі зміщена вбік організації взаємодії «клієнт-сервер». У контексті задачі організації IPC із сервісами дуже важливо й те, що іменовані канали дозволяють скористатися вбудованими можливостями захисту Windows NT.

Таким чином, для організації IPC служби і клієнта при виконанні цієї лабораторної роботи слід скористатися можливостями іменованих каналів і засобами синхронізації, що розглядалися в теоретичних відомостях лабораторних робіт № 6 та № 7 першої частини цих методичних вказівок.

Порядок виконання роботи

1. Модернізувати додаток сервісного процесу попередньої лабораторної роботи, перетворивши його на інформаційно-довідковий сервер. Служба повинна надавати за запитами локальних клієнтів інформацію про поточний системний час. Забезпечити можливість одночасної роботи не більше ніж з трьома клієнтами.
2. Для тестування роботи інформаційно-довідкового сервера розробити додаток клієнта.

Зміст звіту

Звіт з цієї лабораторної роботи повинен містити:

1. Назву і мету лабораторної роботи.
2. Відповіді на контрольні питання.
3. Результати (за необхідності у вигляді Screenshot) отримані при виконанні лабораторної роботи.
4. Повний вихідний текст усіх файлів проектів розроблених та модернізованих додатків з їх детальними коментарями.

Контрольні питання

1. Які властивості відрізняють службу від звичайного додатка?
2. Якими критеріями потрібно користуватися при визначенні того, з правами якого облікового запису буде працювати розроблювана служба?

3. Як запустити службу з правами облікового запису зазначеного користувача?
4. Що таке уособлення для іменованих каналів і навіщо воно може використовуватися службою?

Література: [3, с. 403–434; 7, с. 483–510].

Лабораторна робота № 6

Тема. Мережеві інтерфейси сервісних процесів

Мета: набуття практичних навичок програмної реалізації мережевих інтерфейсів сервісних процесів Windows NT.

Короткі теоретичні відомості

Іменовані канали передбачають можливість організації мережевої взаємодії додатків і при цьому реалізація самих додатків не вимагає від розробника особливих знань механізму роботи мережних протоколів, однак взаємодія таких додатків можлива *тільки у локальних мережах Windows NT*.

Організувати взаємодію із додатками, що працюють під керуванням різних операційних систем можна за допомогою Sockets API. Найбільш часто сокети використовуються для роботи в IP-мережах і їх можна використати для взаємодії додатків не тільки за спеціально розробленими, але й за стандартними протоколами прикладного рівня – HTTP, FTP, Telnet і т. д.

Сокети підтримують множину стандартних мережних протоколів, надають уніфікований інтерфейс для роботи з ними і можуть використовуватися для організації взаємодії додатків на одному комп'ютері, у локальній мережі або через Internet, що дозволяє створювати розподілені додатки різної складності.

Таким чином, для організації взаємодії служби і мережевих клієнтів при виконанні цієї лабораторної роботи необхідно скористатися засобами Sockets API і теоретичними відомостями лабораторної роботи № 2 другої частини цих методичних вказівок.

Порядок виконання роботи

1. Модернізувати додаток сервісного процесу попередніх лабораторних робіт. Служба повинна надавати за запитами як локальних, так і мережевих клієнтів інформацію про поточний системний час. Забезпечити можливість одночасної роботи служби не більше ніж з трьома локальними та трьома мережевими клієнтами.
2. Для тестування роботи інформаційно-довідкового сервера розробити додаток мережевого клієнта.

Зміст звіту

Звіт із цієї лабораторної роботи повинен містити:

1. Назву і мету лабораторної роботи.
2. Відповіді на контрольні питання.
3. Результати (за необхідності у вигляді Screenshot) отримані при виконанні лабораторної роботи.
4. Повний вихідний текст усіх файлів проекту модернізованих і розроблених додатків з їх детальними коментарями.

Контрольні питання

1. Які переваги і недоліки використання сокетів порівняно з іменованими каналами для організації інтерфейсу служби?
2. Що таке експлойт (exploit) і чому це набуває особливої важливості при розробці служби?
3. Що таке (D)DoS-атака і чи можна їй протидіяти?

Література: [3, с. 438–515; 6, с. 17–102].

Лабораторна робота № 7

Тема. Керування параметрами роботи сервісів

Мета: набуття практичних навичок використання реєстру Windows NT для налагодження параметрів роботи сервісних процесів Windows NT.

Короткі теоретичні відомості

Операційна система розглядає сервісні процеси особливим чином і надає їм деяких додаткових функцій. Зокрема від звичайного додатка службу відрізняють такі її властивості, як можливість роботи в довільному контексті безпеки або взагалі запуск служби до реєстрації користувача в системі. Внаслідок цього, особливістю служб є те, що вони не мають користувальницького інтерфейсу.

Користувальницький інтерфейс службі забезпечує програма керування сервісом (SCP), яка, як правило, розробляється при її створенні. SCP є звичайним додатком, який може використовувати всі звичні елементи організації інтерфейсу користувача, зокрема графічні. Програма керування сервісом виступає посередником між користувачем і службою, надаючи, з одного боку, користувачеві звичний для нього інтерфейс, а з іншого боку, взаємодіючи зі службою з використанням відомих тільки розробникові SCP і служби засобів організації IPC.

SCP призначена для керування службою, а також використовується як засіб налагодження або зміни параметрів служби. Для зберігання змінюваних локальних параметрів конфігурації служби, як правило, використовується системний реєстр.

Реєстр – це ієрархічна база даних, що використовується операційною системою як центральне сховище конфігураційних даних. Реєстр забезпечує комплексну взаємодію різних компонентів операційної системи, її додатків та інтерфейсу користувача.

Реєстр має логічну і фізичну структуру. Логічно реєстр нагадує структуру файлової системи, де як імена дискових пристроїв виступають імена визначених кореневих розділів. У таблиці 7.1 наведено опис розділів реєстру на прикладі Windows XP.

Таблиця 7.1 – Опис розділів реєстру

Назва розділу	Опис
HKEY_LOCAL_MACHINE (HKLM)	Набір даних, які стосуються всього комп'ютера в цілому і впливають на всіх користувачів системи
HKEY_CLASSES_ROOT (HKCR)	Псевдонім для HKLM\SOFTWARE\Classes і HKCU\SOFTWARE\Classes. Містить асоціації файлів і реєстрацію COM-компонентів
HKEY_USERS (HKU)	Набір даних, які використовуються при роботі певного користувача. Містить параметри для користувача Default, LocalSystem, LocalService, NetworkService і для користувача із завантаженим у цей момент профілем
HKEY_CURRENT_USER (HKCU)	Псевдонім для HKU\SID поточного користувача
HKEY_CURRENT_CONFIG (HKCC)	Псевдонім для HKLM\SYSTEM\ CurrentControlSet\Hardware Profiles\Current. Містить дані поточного профілю устаткування

Фізично реєстр складається з декількох файлів на диску – вуликів. На диску зберігаються файли (вулики), що складають кореневі розділи HKLM й HKU. Інші кореневі розділи є посиланнями на ключі цих кореневих розділів.

Від кореневих розділів починаються підрозділи – ключі реєстру. Ім'я ключа реєстру не повинне починатися із символу «.» і може містити до 512 символів ANSI або 256 символів Unicode, за винятком символів «\», «*», «?».

Кінцевим елементом в ієрархічній деревоподібній структурі реєстру є значення реєстру. Кожен ключ (підрозділ) реєстру містить як мінімум одне значення, що є значенням за замовчанням (default). За значенням реєстру закріплені такі атрибути, як ім'я і тип. На імена значень накладаються ті ж обмеження, що й на імена ключів. В одному ключі всі імена значень повинні бути унікальними.

Теоретично у реєстрі можуть зберігатися дані 15 типів. Таблиця 7.2 містить опис типів даних, які зустрічаються найбільше часто. Кожне значення може бути порожнім, містити NULL або містити дані. Теоретично дані можуть мати розмір не більш ніж 32 767 байтів.

Таблиця 7.2 – Опис типів даних реєстру

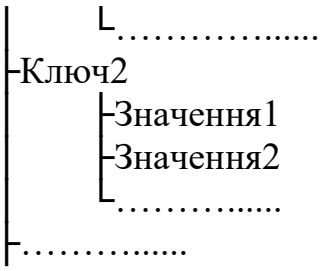
Тип даних	Опис
REG_BINARY	Двійкові дані. Послідовність (масив) байтів
REG_DWORD	Подвійне слово
REG_QWORD	Зчетверене слово
REG_SZ	Рядок, що завершується нулем
REG_MULTI_SZ	Мультирядок – послідовність закритих нулем рядків, що завершується двома нулями
REG_EXPAND_SZ	Рядок, що завершується нулем і містить змінні оточення ОС у вигляді %Ім'я змінної%

Функції для роботи з реєстром *не інтерпретують змінні оточення ОС* виду "%Ім'я змінної%". Перетворення виконується за допомогою функції

```
DWORD ExpandEnvironmentStrings(
    LPCTSTR lpSrc,
    LPTSTR lpDst,
    DWORD nSize)
```

При збереженні параметрів у реєстрі необхідно визначити, кому ця інформація призначена. Додатки та служби зберігають дані, що є *специфічними для конфігурації комп'ютера*, на якому вони виконуються, у наступній ієрархії:

```
HKEY_LOCAL_MACHINE
    └SOFTWARE
        └Компанія
            └Програмний продукт
                └Версія
                    ├Ключ1
                    │   ├Значення1
                    │   ├Значення2
```



Відомості про *специфічні для користувача* параметри конфігурації служби мають таку саму ієрархію, але починається вона від кореневого розділу HKEY_CURRENT_USER.

Для роботи з реєстром використовуються два набори функцій: Reg-функції, експортовані advapi32.dll, та Shell-функції, експортовані shlwapi.dll. Відмінності цих наборів полягають у тому, що використання Shell-функції у деяких випадках виглядає зручнішим. У випадку успішного завершення всі перераховані далі функції повертають ERROR_SUCCESS.

Перед читанням/записом даних реєстру необхідно одержати ідентифікатор розділу реєстру. Ідентифікатор існуючого розділу можна одержати за допомогою функцій:

```
LONG RegOpenKey(
    HKEY        hKey,
    LPCTSTR     lpSubKey,
    PHKEY       phkResult)
```

```
LONG RegOpenKeyEx(
        HKEY        hKey,
        LPCTSTR     lpSubKey,
        DWORD       ulOptions,
        REGSAM      samDesired,
        PHKEY       phkResult)
```

hKey – ідентифікатор раніше відкритого або створеного ключа реєстру, що повертає функція RegCreateKeyEx або RegOpenKeyEx, або одна з констант HKEY_LOCAL_MACHINE, HKEY_CLASSES_ROOT, HKEY_USERS, KEY_CURRENT_CONFIG, HKEY_CURRENT_USER, відповідна одному з однойменних визначених кореневих розділів.

lpSubKey – покажчик на рядок, що закривається нулем і містить ім'я ключа, що відкривається.

ulOptions – зарезервований і повинен бути NULL.

samDesired – маска доступу до ключа, складена з констант:

KEY_CREATE_LINK – дозволено створювати символьні посилання;

KEY_CREATE_SUBKEY – дозволено створювати підрозділи;

KEY_ENUMERATE_SUBKEY – дозволений перебір підрозділів;

KEY_EXECUTE – дозволений доступ для читання;

KEY_NOTIFY – дозволені повідомлення про зміни;

KEY_QUERY_VALUE – дозволені запити даних підрозділів;

KEY_SET_VALUE – дозволено встановлювати дані підрозділів;

KEY_READ – комбінація KEY_ENUMERATE_SUBKEY, KEY_NOTIFY і KEY_QUERY_VALUE;

KEY_WRITE – комбінація KEY_CREATE_SUBKEY і KEY_SET_VALUE;

KEY_ALL_ACCESS – комбінація KEY_QUERY_VALUE, KEY_NOTIFY, KEY_ENUMERATE_SUBKEY, KEY_SET_VALUE, KEY_CREATE_LINK, KEY_CREATE_SUBKEY.

phkResult – адреса змінної для збереження ідентифікатора відкритого розділу у випадку успішного завершення функції.

Одержати ідентифікатор існуючого розділу або створити новий розділ дозволяє функція

```
LONG RegCreateKeyEx(
    HKEY        hKey,
    LPCTSTR     lpSubKey,
    DWORD       Reserved,
    LPTSTR      lpClass,
    DWORD       dwOptions,
    REGSAM      samDesired,
    LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    PHKEY       phkResult,
    LPDWORD     lpdwDisposition)
```

Призначення параметрів hKey, lpSubKey, samDesired, phkResult те саме, що й для функцій RegOpenKey/RegOpenKeyEx, і у випадку якщо розділ існує,

RegCreateKeyEx виконує його відкриття подібно RegOpenKeyEx, а якщо розділ не існує, функція намагається його створити.

Параметр lpdwDisposition є покажчиком на змінну, до якої функція заносить константу REG_CREATED_NEW_KEY, якщо розділ був створений, або константу REG_OPENED_EXISTING_KEY, якщо розділ був відкритий.

Параметри Reserved й lpClass повинні бути NULL, а параметр dwOptions одна з констант:

REG_OPTION_NON_VOLATILE – розділ створюється в реєстрі та зберігається на диску;

REG_OPTION_VOLATILE – розділ створюється у пам’яті та не зберігається при перезавантаженні системи;

REG_OPTION_BACKUP_RESTORE – розділ використовується програмним забезпеченням резервного копіювання для збереження і поновлення реєстру.

lpSecurityAttributes – покажчик на ідентифікатор захисту (ігнорується якщо розділ відкривався).

Додавання значень до реєстру виконується за допомогою функцій:

```
LONG RegSetValueEx(
    HKEY        hKey,
    LPCTSTR     lpValueName,
    DWORD       Reserved,
    DWORD       dwType,
    const BYTE* lpData,
    DWORD       cbData);
```

```
DWORD SHSetValue(
    HKEY        hkey,
    LPCTSTR     pszSubKey,
    LPCTSTR     pszValue,
    DWORD       dwType,
    LPCVOID     pvData,
    DWORD       cbData);
```

hKey – ідентифікатор раніше відкритого або створеного ключа реєстру, що повертає функція RegCreateKeyEx або RegOpenKeyEx або одна з констант HKEY_CLASSES_ROOT,KEY_CURRENT_CONFIG, HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, HKEY_USERS відповідна одному з однойменних визначених кореневих розділів. Ідентифікатор повинен бути отриманий при

виклику функцій RegCreateKeyEx або RegOpenKeyEx з параметром samDesired з установленим прапорцем маски доступу KEY_SET_VALUE.

lpValueName/pszValue – покажчик на рядок з ім'ям установлюваного значення. Якщо значення з таким ім'ям не існує, воно буде створене.

pszSubKey – покажчик на рядок, що закривається нулем і містить ім'я ключа, до якого додається значення.

Reserved – повинно бути 0.

dwType – тип даних (таблиця 7.2).

lpData/pvData – покажчик на буфер, що містить дані для встановлюваного значення з ім'ям, що задане lpValueName.

cbData – розмір даних у байтах у буфері за адресою lpData/pvData.

Одержати значення реєстру можна за допомогою функцій з аналогічними RegSetValueEx/SHSetValue параметрами:

```
LONG RegQueryValueEx(
    HKEY    hKey,
    LPCTSTR lpValueName,
    LPDWORD lpReserved,
    LPDWORD lpType,
    LPBYTE  lpData,
    LPDWORD lpcbData);
```

```
DWORD SHGetValue(
    HKEY    hkey,
    LPCTSTR pszSubKey,
    LPCTSTR pszValue,
    LPDWORD pdwType,
    LPVOID  pvData,
    LPDWORD pcbData);
```

Порядок виконання роботи

1. Модернізувати додаток сервісного процесу попередніх лабораторних робіт. При старті служба повинна отримувати параметри (ім’я іменованого каналу, номер порту, IP-адресу, ім’я і місце розташування log-файлу, формат дати/часу тощо) з реєстру Windows. Передбачити можливість динамічної (тобто під час роботи служби) зміни параметрів роботи служби.
2. Для тестування роботи служби розробити програму керування службою, яка дозволяє:

   а) установлювати службу (з передачею початкових параметрів роботи);

б) призупиняти/поновлювати/зупиняти/запускати службу;

в) передавати службі будь-які команди;

г) видаляти службу;

д) показувати поточний стан і параметри роботи служби;

е) змінювати параметри роботи служби.

Зміст звіту

Звіт із цієї лабораторної роботи повинен містити:

1. Назву і мету лабораторної роботи.
2. Відповіді на контрольні питання.
3. Результати (за необхідності у вигляді Screenshot), отримані при виконанні лабораторної роботи.
4. Повний вихідний текст усіх файлів проектів модернізованих і розроблених додатків з їх детальними коментарями.

Контрольні питання

1. Що необхідно зробити для забезпечення можливості взаємодії служби з елементами робочого столу користувача?
2. Які переваги надає збереження параметрів служби у реєстрі ОС?
3. Якими критеріями потрібно користуватися при визначенні місця збереження параметрів служби у реєстрі?
4. Чи може декілька служб зберігати свої параметри в одній і тій самій гілці реєстру ОС?
5. Чи існує обмеження на кількість та/або розмір параметрів служби у реєстрі ОС?

Література: [3, с. 136–174; 8, с. 26–58].

2 КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ СТУДЕНТАМИ

У 6-му семестрі студенти виконують 9 лабораторних робіт. Загальна кількість балів, яку отримують студенти за виконання та захист усіх лабораторних робіт складає 54 бали (6 балів за кожну лабораторну). У цій частини методичних вказівок наведено 7 лабораторних робіт 6-го семестру. Отже за виконання та захист усіх лабораторних робіт цієї частини методичних вказівок студенти отримують 42 бали. Інші 2 лабораторні роботи наведені в попередній частини цих методичних вказівок.

Шкала оцінювання: національна та ECTS

<table>
<tr><td rowspan="2">Сума балів за всі види навчальної діяльності</td><td rowspan="2">Оцінка ECTS</td><td colspan="2">Оцінка за національною шкалою</td></tr>
<tr><td>Для екзамену, курсового проекту (роботи), практики</td><td>Для заліку</td></tr>
<tr><td>90–100</td><td>A</td><td>Відмінно</td><td rowspan="5">Зараховано</td></tr>
<tr><td>82–89</td><td>B</td><td rowspan="2">Добре</td></tr>
<tr><td>74–81</td><td>C</td></tr>
<tr><td>64–73</td><td>D</td><td rowspan="2">Задовільно</td></tr>
<tr><td>60–63</td><td>E</td></tr>
<tr><td>35–59</td><td>FX</td><td>Незадовільно з можливістю повторного складання</td><td>Не зараховано з можливістю повторного складання</td></tr>
<tr><td>0–34</td><td>F</td><td>Незадовільно з обов’язковим повторним вивченням навчальної дисципліни</td><td>Не зараховано з обов’язковим повторним вивченням навчальної дисципліни</td></tr>
</table>

СПИСОК ЛІТЕРАТУРИ

1. Зайцев О. В. Rootkits, Spyware / Adware, Keyloggers & Backdoors: обнаружение и защита / О. В. Зайцев. – СПб. : БХВ-Петербург, 2006. – 304 с. : ил.
2. Побегайло А. П. Системное программирование в Windows / А. П. Побегайло. – СПб. : БХВ-Петербург, 2006. – 1056 с. : ил.
3. Рихтер Дж. Программирование серверных приложений для Microsoft Windows 2000. Мастер-класс / Дж. Рихтер, Дж. Д. Кларк; пер. с англ. – СПб. : Питер; М. : Издательско-торговый дом «Русская Редакция», 2001. – 592 с. : ил.
4. Рихтер Дж. Windows для профессионалов: создание эффективных Win32 приложений с учетом специфики 64-разрядной версии Windows / Дж. Рихтер; пер. с англ. − [4-е изд.]. − СПб. : Питер; М. : Издательско-торговый дом "Русская Редакция", 2004. − 749 с. : ил.
5. Румянцев П. В. Работа с файлами в Win 32 API / П. В. Румянцев. – [2-е изд.]. – М. : Горячая линия-телеком, 2002. – 216 с.
6. Фостер Дж. Разработка средств безопасности и эксплойтов / Дж. Фостер, В. Лю; пер. с англ. – М. : Русская редакция; СПб. : Питер, 2007. – 749 с. : ил.
7. Харт Д. М. Системное программирование в среде Windows / Д. М. Харт; пер. с англ. – [3-е изд.]. – М : Издательский дом «Вильямс», 2005. – 592 с. : ил.
8. Хонейкатт Дж. Реестр Microsoft Windows XP. Справочник профессионала : практическое пособие / Дж. Хонейкатт; пер. с англ. – М. : Издательство «СП ЭКОМ», 2003. – 656 с. : ил.

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Системне програмне забезпечення» для студентів денної та заочної форм навчання зі спеціальності 123 – «Комп'ютерна інженерія». Частина ІІІ.

Укладачі: старш. викл. Ю. В. Зілінський,
асист. Р. С. Цвєнтарний

Відповідальний за випуск в.о. зав. кафедри КІС проф. М. І. Гученко

Підп. до др. _____ Формат 60х84 1/16. Папір тип. Друк ризографія.
Ум. друк. арк. ___. Наклад _____ прим. Зам. № _____. Безкоштовно.

Видавничий відділ
Кременчуцького національного університету
імені Михайла Остроградського
вул. Першотравнева, 20, м. Кременчук, 39600