МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

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

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 І.

Укладачі: старш. викл. Ю.В. Зілінський,
старш. викл. О.М. Мотолига
Рецензент 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 й при самостійному читанні спеціальної літератури, та 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.

Як засіб розробки при виконанні лабораторних робіт використовується безкоштовно поширюваний пакет асемблера MASM32 v10 (http://www.masm32.com) та інтеґроване середовище розробки програм RadAsm (http://radasm.visualassembler.com), яке також розповсюджується безкоштовно, і може використовуватися з пакетами асемблерів MASM, TASM, FASM, NASM, HLA, GoASM та С-компіляторами LCC, Borland C++, Microsoft Visual C++. Як засіб налагодження використовується поширюваний безкоштовно наладчик OllyDbg 1.10 (http://home.t-online.de/home/Ollydbg). Він не є наладчиком режиму ядра, але його можливостей досить для налагодження більшості додатків.

Тематика лабораторних робіт даних методичних вказівок перетинається з тематикою практичних занять з курсу та охоплює традиційні для системного програмування питання:

- керування файловими системами;
- керування пам'яттю, процесами й потоками;
- організація взаємодії поміж процесами;
- планування й синхронізація процесів і потоків;
- структурне й векторне оброблення виключень;
- перехоплення системних викликів;
- розроблення системних сервісів Windows NT;
- робота з реєстром.

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 в позааудиторний час. Підготовка складається з вивчення відповідних розділів лекційного матеріалу, коротких теоретичних відомостей, що наведені як безпосередньо в 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 із 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 в 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 якого 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.

Перша частина методичних вказівок охоплює сім робіт, передбачених для виконання у п’ятому навчальному семестрі робочою навчальною програмою дисципліни «Системне програмне забезпечення» для студентів денної та заочної форм навчання зі спеціальності 123 – «Комп'ютерна інженерія» (у тому числі скорочений термін навчання).

1 ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

Лабораторна робота № 1

Тема. Пошук файлу/каталогу й перегляд вмісту каталогу.

Мета: набуття практичних навичок пошуку й перегляду вмісту каталогів та організації виконання операцій введення за допомогою командного рядка програми.

Короткі теоретичні відомості

Перегляд вмісту каталогу (пошук або перевірка існування файлу/каталогу) виконується за допомогою трьох функцій: FindFirstFile, FindNextFile, FindClose. Після виклику функції FindFirstFile (залежно від результатів її виклику) пошук здійснюється викликом у циклі функції FindNextFile. Після завершення циклу необхідно закрити сеанс пошуку за допомогою функції FindClose.

```
HANDLE FindFirstFile(
  LPCTSTR                lpFileName,
  LPWIN32_FIND_DATA      lpFindFileData);
```

Через параметр lpFileName передається адреса рядка, що містить шлях до каталогу й шаблон для пошуку. У шаблоні можна використати символи “?” й “*”. Через параметр lpFindFileData передається адреса структури типу WIN32_FIND_DATA, до якої буде записана інформація про знайдені файли. Ця структура визначена в такий спосіб:

```
typedef struct _WIN32_FIND_DATA  {
  DWORD        dwFileAttributes;          // атрибути файла
  FILETIME     ftCreationTime;            // час створення файла
  FILETIME     ftLastAccessTime;          // час доступу
  FILETIME     ftLastWriteTime;           // час модифікації
  DWORD        nFileSizeHigh;             // розмір файла (старше слово)
```

```
  DWORD        nFileSizeLow;              // розмір файла (молодше слово)
  DWORD        dwReserved0;               // зарезервовано
  DWORD        dwReserved1;          // зарезервовано
  TCHAR    cFileName[MAX_PATH];      // ім'я файла
  TCHAR    cAlternateFileName[14];       // альтернативне (8.3) ім'я файла
} WIN32_FIND_DATA;
```

Якщо пошук завершився успішно, функція FindFirstFile повертає ідентифікатор (хендл) пошуку, що потім використається в циклі при виклику функції FindNextFile. При помилці повертається значення INVALID_HANDLE_VALUE (-1, 0FFFFFFFFh).

Якщо виклик FindFirstFile завершився успішно, в циклі виконується виклик функції FindNextFile:

```
BOOL FindNextFile(
  HANDLE hFindFile,
  LPWIN32_FIND_DATA lpFindFileData);
```

Через параметр hFindFile цієї функції потрібно передати ідентифікатор пошуку, отриманий від функції FindFirstFile. Через параметр lpFindFileData передається адреса тієї ж самої структури типу WIN32_FIND_DATA, що була використана при виклику функції FindFirstFile.

Якщо функція FindNextFile завершилася успішно, вона повертає значення TRUE. При помилці повертається значення FALSE.

Після завершення циклу перегляду необхідно закрити ідентифікатор пошуку, викликавши для цього функцію BOOL FindClose(HANDLE hFindFile).

Як єдиний параметр цієї функції передається ідентифікатор пошуку, отриманий від функції FindFirstFile.

Атрибути файла/каталогу dwFileAttributes у структурі типу WIN32_FIND_DATA являють собою диз’юнкцію наступних констант:

```
FILE_ATTRIBUTE_READONLY    equ 1h – тільки для читання
FILE_ATTRIBUTE_HIDDEN      equ 2h – прихований
FILE_ATTRIBUTE_SYSTEM      equ 4h – системний
```

FILE_ATTRIBUTE_DIRECTORY equ 10h – каталог

FILE_ATTRIBUTE_ARCHIVE equ 20h – архівний

FILE_ATTRIBUTE_NORMAL equ 80h – немає жодних атрибутів

FILE_ATTRIBUTE_TEMPORARY equ 100h – тимчасовий

FILE_ATTRIBUTE_COMPRESSED equ 800h – стиснутий

FILE_ATTRIBUTE_ENCRYPTED equ 4000h – шифрований

Час створення, доступу та модифікації файла/каталогу в структурі типу WIN32_FIND_DATA має тип FILETIME, який визначається структурою:

```
typedef struct _FILETIME {
   DWORD dwLowDateTime;
   DWORD dwHighDateTime;
} FILETIME, *PFILETIME, *LPFILETIME;
```

поля якої містять молодше й старше подвійне слово 64-бітного формату UTC – кількість інтервалів по 100 наносекунд (10^{-9} с), що пройшли з 0 годин 0 хвилин 0 секунд 1 січня 1601 року за годинним поясом Гринвіча.

Для перетворення з 64-бітного формату в звичний DD/MM/YYYY HH:MM:SS використовується функція

```
BOOL FileTimeToSystemTime(
  const FILETIME* lpFileTime,        // адреса структури типу FILETIME
  LPSYSTEMTIME lpSystemTime);  // адреса структури типу SYSTEMTIME
```

lpFileTime – адреса структури типу FILETIME, яка містить дату/час для перетворення, а lpSystemTime – адреса структури типу SYSTEMTIME, у яку повертається перетворений час. SYSTEMTIME визначається наступним чином:

```
typedef struct _SYSTEMTIME {
WORD wYear;              // рік (1601-30827)
WORD wMonth;             // місяць
WORD wDayOfWeek;  // день тижня (0 – неділя, 1 – понеділок і.т.д.)
WORD wDay;               // день місяця
WORD wHour;              // година (0-23)
WORD wMinute;            // хвилина
```

```
WORD wSecond;          // секунда
WORD wMilliseconds;    // мілісекунда
} SYSTEMTIME, *PSYSTEMTIME;
```

Для обробки параметрів командного рядка програми використовується бібліотечна функція MASM32 DWORD GetCL(DWORD ArgNum:DWORD, LPCSTR ItemBuffer). Якщо функція завершилася успішно, вона повертає значення 1 і рядок з параметром номер ArgNum (0 – повне ім'я програмного файла) у буфер за адресою ItemBuffer.

Для перетворення з машинного подання числових даних у символьний вигляд для виведення на консоль використовується бібліотечна функція MASM32 VOID dwtoa(DWORD dwValue, LPCSTR lpBuffer). Ця функція повертає в буфер за адресою lpBuffer рядок із символьним поданням значення dwValue в десятковій системі числення з урахуванням знака.

У полі cFileName структури WIN32_FIND_DATA рядок з іменем файла/каталогу подається у кодуванні, що використовується в додатках з графічним інтерфейсом користувача. Для символів національних алфавітів це кодування відрізняється від кодування, що використовується в консольних додатках. Для коректного відображення при виведенні на консоль імен файлів/каталогів, у яких можуть зустрічатися символи, відмінні від латиниці, попередньо виконується їх перекодування за допомогою функції BOOL CharToOem(LPCTSTR lpszSrc, LPSTR lpszDst).

Для виведення на консоль використовується бібліотечна функція MASM32 VOID StdOut(LPSTR lpszText). Ця функція направляє в поточний потік виведення, починаючи з поточної позиції курсора рядок символів, що завершується нульовим байтом, і переміщує курсор у позицію за останнім виведеним символом.

Порядок виконання роботи

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

2. Відкрити каталог lst і виконати запуск додатка lst.exe з наступними параметрами командного рядка:
   а) lst.exe;
   б) lst.exe %SystemRoot%*.*;
   в) lst.exe %SystemRoot%.
3. Зазначити в звіті, яким чином додаток lst.exe сприймає різні варіанти параметрів командного рядка
4. Відкрити файл проекту lst.rap.
5. Виконати трансляцію проекту. За результатами трансляції виправити три синтаксичні помилки. Занести до звіту помилкові рядки, виправлені рядки та пояснення щодо виправлених помилок.
6. Виконати лінкування проекту та одержати програмний файл lst.exe.
7. Повторити п. 1 завдання, і, якщо робота lst.exe отриманого при виконанні п. 5 відрізняється від його роботи в п. 1, то виправити ***логічні*** помилки в проекті. Занести до звіту помилкові рядки, виправлені рядки та пояснення щодо виправлених помилок.
8. Модернізувати додаток lst.exe таким чином, щоб виклик, наприклад, lst.exe %SystemRoot% сприймався і виконувався так само, як і lst.exe %SystemRoot%*.*. Тобто, якщо як параметр командного рядка зазначено ім’я каталогу, то додаток повинен виводити на консоль інформацію про всі файли цього каталогу.
9. Модернізувати додаток lst.exe таким чином, щоб він за наявності третього параметра командного рядка «/h» брав до уваги тільки ті файли, що мають установлений атрибут «прихований», а за відсутності цього параметра – працював, як і раніше.

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 та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проекту модернізованого додатка з його детальними коментарями.

Контрольні питання

1. Яку з файлових систем не підтримує Windows NT:
   а) UDF;
   б) Ext2/Ext3;
   в) FAT;
   г) NTFS.
2. Зазначити правильні твердження щодо викликів функцій Win32 API
   а) функція, що викликається, одержує всі параметри через стек;
   б) усі параметри являють собою 32-розрядні значення;
   в) параметри не можуть задаватися безпосередніми значеннями;
   г) повернення стека у вихідний стан після виклику виконує функція, що викликається;
   д) функція Win32 API, що викликається, гарантовано зберігає регістри загального призначення ESI, EDI, EBX, EBP;
   е) усі параметри функції завантажуються до стека від першого до останнього, або зліва направо;
   ж) регістр EAX після виклику Win32 API функції містить значення, що повертається;
   и) функція, що викликається, одержує перші чотири параметри в регістрах ESI, EDI, EBX, EBP, а інші через стек;
   к) усі параметри функції завантажуються до стека від першого до останнього, або зліва направо;
   л) усі параметри функції завантажуються до стека від останнього до першого, або справа наліво;

м) повернення стека у вихідний стан після виклику виконує додаток, що викликає функцію.

3. Геометрію логічного пристрою (кількість байтів у секторі, кількість секторів у кластері, загальну кількість кластерів) можна визначити за допомогою Win API функції:
   и) GetLogicalDriveStrings;
   к) GetDriveType;
   л) GetDiskFreeSpace;
   м) GetVolumeInformation.
4. Розмір файла у Windows NT визначається числом завдовжки:
   и) 16 біт;
   к) 32 біти;
   л) 64 біти;
   м) 128 біт.
5. Відмітки часу створення, модифікації, останнього доступу до файла Windows NT зберігає у вигляді:
   и) 16-бітного числа, що дорівнює кількості секунд, які пройшли з 0 годин 0 хвилин 0 секунд 1 січня 1980 року за годинним поясом Гринвіча;
   к) 32-бітного числа, що дорівнює кількості секунд, які пройшли з 0 годин 0 хвилин 0 секунд 1 січня 1970 року за годинним поясом Гринвіча;
   л) 64-бітного числа, що дорівнює кількості інтервалів по 100 наносекунд, які пройшли з 0 годин 0 хвилин 0 секунд 1 січня 1601 року за годинним поясом Гринвіча;
   м) структури типу SYSTEMTIME.

Література: *[3, с. 25 – 46; 5, Глава 10, с.649 – 661; 6, Глава 2, с.51 – 54; 7, Глава 5].*

Лабораторна робота № 2

Тема. Основний API введення/виведення і робота з файлами.

Мета: набуття практичних навичок виконання основних операцій файлового введення/виведення.

Короткі теоретичні відомості

Під основним API тут розуміють ті функції, які дозволяють керувати файловою системою й отримувати доступ до вмісту файлів. Найперша з них це функція CreateFile, опис усіх параметрів якої займе з десяток сторінок. Тут будуть наведені тільки описи прототипів набору основних функцій для роботи з файлами з коротким описом параметрів. За детальним описом усіх параметрів варто звертатися до літератури або MSDN, хоча в багатьох випадках призначення того або іншого параметра цілком очевидне.

За допомогою функції CreateFile можна виконувати створення нового файла, відкриття існуючого файла або каталогу, зміну довжини існуючого файла, відкриття фізичного диска, тому, консолі, послідовного порту, каналу, поштового слоту, пристрою.

```
HANDLE CreateFile(
    LPCTSTR lpFileName,             // адреса рядка імені файла
    DWORD  dwDesiredAccess,         // режим доступу
    DWORD  dwShareMode,             // режим спільного доступу до файла
    LPSECURITY_ATTRIBUTES lpSecurityAttributes, // дескриптор захисту
    DWORD  dwCreationDistribution,  // параметри створення
    DWORD  dwFlagsAndAttributes,    // атрибути файла
    HANDLE hTemplateFile);          // ідентифікатор файла з атрибутами
```

За допомогою функцій ReadFile й WriteFile додаток може виконувати, відповідно, читання з файла й запис до файла.

```
BOOL ReadFile(
    HANDLE hFile,                   // ідентифікатор файла
    LPVOID lpBuffer,                // адреса буфера для даних
```

```
    DWORD nNumberOfBytesToRead,      // кількість байт для читання
    LPDWORD lpNumberOfBytesRead,     // адреса слова, у яке буде записано
                                     // кількість прочитаних байт
    LPOVERLAPPED lpOverlapped);// адреса структури типу OVERLAPPED
BOOL WriteFile(
    HANDLE hFile,                    // ідентифікатор файла
    LPVOID lpBuffer,                 // адреса записуваного блоку даних
    DWORD nNumberOfBytesToWrite,     // кількість байт для запису
    LPDWORD lpNumberOfBytesWrite,    // адреса слова, у якому буде
                                     //  збережена кількість записаних байт
    LPOVERLAPPED lpOverlapped); // адреса структури типу OVERLAPPED
```

Функція SetFilePointer дозволяє виконати установку поточної файлової позиції для виконання операції введення/виведення:

```
DWORD SetFilePointer(
    HANDLE hFile,                // ідентифікатор файла
    LONG lDistanceToMove,        // кількість байт, на яку буде
                                 // пересунена поточна позиція
    PLONG lpDistanceToMoveHigh,  // адреса старшого слова, що містить
                                 // відстань для переміщення позиції
    DWORD dwMoveMethod);         // спосіб переміщення позиції
```

Функція

```
    DWORD GetFileSize(
    HANDLE hFile,                // ідентифікатор файла
    LPDWORD lpFileSizeHigh);     // адреса старшого слова для розміру файла
```

повертає молодше 32-розрядне слово 64-розрядного розміру файла з ідентифікатором hFile. Старше слово розміру файла записується в змінну типу DWORD, адреса якої передається функції через параметр lpFileSizeHigh. При помилці GetFileSize повертає -1 (INVALID_VALUE_HANDLE)

Визначити атрибути файла можна за допомогою функції

```
DWORD GetFileAttributes(LPCTSTR lpFileName);
```

Установити нові атрибути можна за допомогою функції

```
BOOL SetFileAttributes(
    LPCTSTR lpFileName,          // адреса рядка шляху до файла
    DWORD dwFileAttributes);     // адреса слова з новими атрибутами.
```

Порядок виконання роботи

1. Завантажити із сервера кафедри архів з файлами поточної лабораторної роботи й розкрити його в будь-який каталог свого облікового запису.
2. Відкрити каталог проекту CatFile.
3. Передивитися вміст файлів file1.txt та file2.txt в каталозі проекту.
4. Виконати запуск додатка CatFile.exe з наступними параметрами:
   а) CatFile.exe;
   б) CatFile.exe file1.txt;
   в) CatFile.exe file2.txt;
   г) CatFile.exe file1.txt file2.txt.
5. Зазначити у звіті, яким чином додаток CatFile.exe сприймає різні варіанти параметрів командного рядка.
6. Передивитися вміст файлів file1.txt і file2.txt.
7. Зазначити у звіті функції додатку CatFile.exe.
8. Відкрити файл проекту CatFile.rap.
9. Виконати трансляцію і лінкування проекту. Виправити за необхідності помилки трансляції та лінкування та одержати програмний файл CatFile.exe.
10. Переконатися, що робота отриманого в попередньому пункті додатка, відповідає його функціям, зазначеним раніше в п. 4 та п. 6. Якщо робота CatFile.exe, отриманого при виконанні п. 8, відрізняється від того, що було зазначено раніше в п. 4 в та п. 6, то виправити логічні помилки в проекті.

11. Виконати запуск додатка CatFile.exe file1.txt file3.txt, тобто за умови, що другий файл (джерело) не існує. Занести до звіту пояснення роботи програми.
12. Виконати запуск додатка CatFile.exe file3.txt file2.txt, тобто за умови, що перший файл (приймач) не існує. Занести до звіту пояснення роботи програми.
13. Модернізувати додаток CatFile.exe таким чином, щоб виклик за умови, що перший файл (приймач) не існує призводив до копіювання другого файла (джерела) у перший файл (приймач).
14. Модернізувати додаток CatFile.exe таким чином, щоб він виконував свої функції за умови, якщо і першим, і другим параметром командного рядка зазначено одне і те саме ім’я.

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 та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проекту модернізованого додатка з його детальними коментарями.

Контрольні питання

1. Основний API введення/виведення експортується системною DLL-бібліотекою:

   а) hal.dll;

   б) user32.dll;

   в) kernel32.dll;

г) gdi32.dll.

2. Перевага використання раннього зв’язування з DLL порівняно з пізнім полягає в тому, що:

   а) для лікування програми потрібно використовувати файли бібліотек імпорту і заголовні файли;

   б) на момент запуску програми всі необхідні DLL виявляються завантаженими автоматично без участі програми;

   в) неможливо завантажити на виконання програму за відсутності DLL, з якою вона була пов’язана при лінкуванні;

   г) DLL завантажується в адресний простір ядра ОС і її код виконується в нульовому кільці захисту.

3. Перевага використання пізнього зв’язування з DLL порівняно з раннім полягає в тому, що:

   а) програма за необхідності може вивантажити непотрібну їй DLL, звільнивши використовувану пам’ять;

   б) неможливо завантажити на виконання програму за відсутності DLL, з якою вона була пов’язана при лінкуванні;

   в) програма за необхідності може скористатися необмеженою кількістю DLL;

   г) на момент запуску програми всі необхідні DLL виявляються завантаженими автоматично без участі програми.

4. За допомогою функції CreateFile неможливо виконати:

   а) відкриття існуючого файла;

   б) відкриття файла, відображеного на пам’ять;

   в) створення нового файла;

   г) зміну довжини існуючого файла;

д) відкриття фізичного диска;

е) відкриття логічного тому;

ж) відкриття мережевого порту;

и) відкриття послідовного порту;

к) відкриття існуючого каталогу;

л) відкриття каналу та поштового слоту;

м) створення нового каталогу;

н) відкриття пристрою.

5. Якщо значення лічильника посилань на об'єкт «файл», стало дорівнювати нулю, то:

и) об'єкт буде зруйнований;

к) доступ до файла буде заборонений;

л) файл буде знищений;

м) файл буде закритий.

Література: *[1, Глава 5, с. 228 – 234; 2, Глава 3, с. 28 – 33; 3, с. 47 – 61; 6, Глава 2, с. 54 – 61; 6, Глава 5].*

Лабораторна робота № 3

Тема. Файли, що відображені в пам’ять.

Мета: набуття практичних навичок використання в програмах файлів, що відображені в пам’ять.

Короткі теоретичні відомості

Як і віртуальна пам'ять, відображувані на пам'ять файли дозволяють резервувати регіон адресного простору й передавати йому фізичну пам'ять. Різниця між цими механізмами полягає в тому, що в останньому випадку

фізична пам'ять не виділяється зі сторінкового файла, а береться з файла, що вже перебуває на диску. Як тільки файл відображено на пам’ять, до нього можна звертатися так, начебто він цілком у неї завантажений.

У процесі відображення адресного простору пам'ять не виділяється, а тільки резервується, і при цьому файли віртуальної пам'яті взагалі не використовуються, тому що сторінки фрагмента зв'язуються з відображуваним файлом.

Для файла відображеного на пам’ять, операційна система забезпечує тотожність вмісту дискового файла й області пам'яті, на яку він відображений, виконуючи за необхідності операції читання й запису дискового файла.

Для використання файла спроектованого на пам'ять, потрібно виконати три операції:

а) створити або відкрити дисковий файл, що буде використовуватися для відображення на пам’ять;

б) створити проекцію файла, щоб повідомити системі розмір файла й спосіб доступу до нього;

в) виконати відображення, указавши системі, як спроектувати в адресний простір процесу файл – цілком або частково.

Робота з файлом, відображеним на пам'ять, виконується як з віртуальною пам'яттю (так, як це було б із пам'яттю, отриманою за допомогою VirtualAlloc). Закінчивши роботу із відображеним на пам’ять файлом, необхідно виконати теж три операції:

а) повідомити систему про скасування проектування на адресний простір процесу проекції файла;

б) закрити проекцію файла;

в) закрити файл.

Для виконання всіх перерахованих вище операцій у програмному інтерфейсі Windows NT є відповідні засоби.

Створення або відкриття дискового файла виконується за допомогою функції CreateFile. Одержавши від функції CreateFile ідентифікатор, необхідно

створити об'єкт «відображення файла». Для створення відображення використовується функція CreateFileMapping, прототип якої наведений нижче:

```
HANDLE CreateFileMapping(
    HANDLE hFile,                    // ідентифікатор відображуваного файла
    LPSECURITY_ATTRIBUTES lpFileMappingAttributes,// дескриптор
                                                  // захисту
    DWORD flProtect,                 // захист для відображуваного файла
    DWORD dwMaximumSizeHigh,         // розмір файла (старше слово)
    DWORD dwMaximumSizeLow,          // розмір файла (молодше слово)
    LPCTSTR lpName);                 // ім'я відображеного файла
```

Одержавши від функції CreateFileMapping ідентифікатор «об'єкта відображення», необхідно виконати саме відображення, викликавши для цього функцію MapViewOfFile (або MapViewOfFileEx), прототип якої наведений нижче

```
LPVOID MapViewOfFile(
    HANDLE hFileMappingObject,// ідентифікатор відображення
    DWORD dwDesiredAccess,    // режим доступу
    DWORD dwFileOffsetHigh,   // зсув у файлі (старше слово)
    DWORD dwFileOffsetLow,    // зсув у файлі (молодше слово)
    DWORD dwNumberOfBytesToMap);// кількість відображуваних байт.
```

У випадку успішного виконання відображення функція MapViewOfFile повертає адресу відображеної області пам'яті, при помилці повертається значення NULL (перевіряти обов'язково, запис/читання за нульовою адресою – виключення!).

Якщо створене відображення більше не потрібно, його необхідно відмінити за допомогою функції BOOL UnmapViewOfFile(LPVOID lpBaseAddress). Через єдиний параметр цієї функції необхідно передати адресу області відображення, отриману від функцій MapViewOfFile.

Далі необхідно закрити ідентифікатори відображення файла й файла отриманих при виклику CreateFileMapping й CreateFile.

Порядок виконання роботи

1. Завантажити із сервера кафедри архів з файлами поточної лабораторної роботи й розкрити його в будь-який каталог свого облікового запису.
2. Відкрити каталог проекту CopyMMF.
3. Передивитися вміст файлів file1.txt та file3.txt у каталозі проекту.
4. Виконати запуск додатка CopyMMF.exe з наступними параметрами:
   а) CopyMMF.exe;
   б) CopyMMF.exe file1.txt;
   в) CopyMMF.exe file1.txt file2.txt;
   г) CopyMMF.exe file1.txt file3.txt.
5. Зазначити у звіті, яким чином додаток CopyMMF.exe сприймає різні варіанти параметрів командного рядка.
6. Передивитися вміст файлів file1.txt, file2.txt, file3.txt
7. Зазначити у звіті функції додатка CopyMMF.exe.
8. Відкрити файл проекту CopyMMF.rap.
9. Виконати трансляцію і лінкування проекту та одержати програмний файл CopyMMF.exe.
10. Переконатися, що робота отриманого в попередньому пункті додатка, відповідає його функціям, зазначеним раніше в п. 6 і за необхідності виправити логічні помилки в додатку.
11. Модернізувати додаток CopyMMF.exe таким чином, щоб у випадку якщо файл, зазначений у другому параметрі командного рядка не існує, додаток створював його і виконував копіювання в нього вмісту файла, зазначеного у першому параметрі командного рядка. У випадку, якщо файл зазначений у другому параметрі командного рядка вже існує, додаток повинен виконувати конкатенацію цього файла з файлом, зазначеним у першому параметрі командного рядка (використовуючи файли, відображені в пам’яті).
12. Занести до звіту вихідний текст модернізованого додатка CopyMMF.exe з його детальними коментарями.

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 та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проекту модернізованого додатка з його детальними коментарями.

Контрольні питання

1. Зазначити неправильне твердження щодо файлів відображених на пам’ять в ОС Windows NT:
   а) при відображенні існуючого файла взагалі не використовується дискова віртуальна пам’ять;
   б) вміст файла, відображеного на пам’ять, не може змінюватися процесом, який його створив;
   в) один і той самий файл або його фрагмент може відображатися на пам’ять лише раз;
   г) відображення файла на пам’ять може виконуватися окремими його частинами.
2. Буфер для зберігання даних фрагменту файла, відображеного на пам’ять в ОС Windows NT:
   а) розташований в адресному просторі додатка користувача, що виконав відображення;
   б) розташований в адресному просторі ядра;
   в) залежно від способу відображення може розташовуватися як в адресному просторі додатка користувача, так й в адресному просторі ядра;
   г) розташований у файлі дискової віртуальної пам’яті та підкачується в адресний простір додатка користувача.

3. Зазначити неправильне твердження щодо організації стандартних куп процесів в ОС Windows NT:
   а) стандартну купу в адресному просторі процесу створює сама система при ініціалізації процесу;
   б) при створенні стандартної купи, її розмір становить 1 Мб і система не дозволяє збільшувати цей розмір;
   в) стандартна купа процесу використовується багатьма функціям Win API;
   г) синхронізацію потоків при доступі до стандартної купи процесу виконує сама система.
4. Зазначити неправильне твердження щодо організації куп процесів в ОС Windows NT:
   а) області пам'яті, що розподіляється динамічно, дозволяють ігнорувати гранулярність виділення пам'яті;
   б) виділення й звільнення пам'яті в купі відбувається повільніше, ніж при використанні віртуальної пам'яті;
   в) синхронізація потоків при доступі до додаткової купи процесу покладається виключно на процес що її створив;
   г) розмір додаткової купи процесу зазначений при її створенні може бути в подальшому збільшений або зменшений.

Література: *[2, Глава 17, с. 409 – 450; 3, с. 82 – 92; 6, Глава 5, с. 155 – 182; 8, Глава 1].*

Лабораторна робота № 4

Тема. Одержання інформації про процеси та потоки системи.

Мета: набуття практичних навичок отримання інформації про стан системи.

Короткі теоретичні відомості

Кількість і стан процесів, потоків, модулів і куп у системі постійно змінюється, і тому при описі функцій для отримання стану системи оперують

таким поняттям, як Snapshot (знімок, фотографія), під яким розуміють інформацію про стан об'єктів системи, зафіксовану на певний момент часу.

Для одержання знімка стану об'єктів системи використовується функція CreateToolhelp32Snapshot(DWORD dwFlags, DWORD th32ProcessID).

Параметр dwFlags вказує, яку інформацію необхідно включити до знімка:

TH32CS_SNAPHEAPLIST = 1 - включити до знімка список куп;

TH32CS_SNAPPROCESS = 2 - включити до знімка список процесів;

TH32CS_SNAPTHREAD = 4 - включити до знімка список потоків;

TH32CS_SNAPMODULE = 8 - включити до знімка список модулів;

TH32CS_SNAPALL = 15 - включити до знімка все перераховане вище.

Через th32ProcessID передається PID процесу, для якого буде зроблений знімок. Цей параметр використовується тільки з dwFlags який дорівнює TH32CS_SNAPHEAPLIST, TH32CS_SNAPMODULE й TH32CS_SNAPALL (для поточного процесу він повинен бути 0), у комбінації з іншими значеннями dwFlags він іґнорується й у знімок, включається інформація про всі процеси (всі потоки всіх процесів).

Функція повертає хендл поточного знімка або -1 (INVAID_VALUE_HANDLE), якщо виклик не вдався.

Подальше добування зі знімка процесів потрібної інформації нагадує пошук файлів за допомогою FindFirstFile й FindNextFile. Для одержання інформації про процеси використовується пара функцій Process32First(HANDLE hSnapshot, LPPROCESSENTRY32W lppe) та Process32Next(HANDLE hSnapshot, LPPROCESSENTRY32 lppe).

Обидві функції при успішному завершенні (TRUE) заповнюють структуру PROCESSENTRY32 (елемент dwSize якої, необхідно заповнити до першого виклику) наступного типу:

```
PROCESSENTRY32 STRUCT
   dwSize                DWORD ?  ; розмір цієї структури
   cntUsage              DWORD ?  ; застаріло, завжди 0
   th32ProcessID         DWORD ?  ; PID
```

```
th32DefaultHeapID       DWORD ?  ; застаріло, завжди 0
th32ModuleID            DWORD ?  ; застаріло, завжди 0
cntThreads              DWORD ?  ; кількість потоків
th32ParentProcessID     DWORD ?  ; PID процесу-батька
pcPriClassBase          DWORD ?  ; базовий пріоритет
dwFlags                 DWORD ?  ; застаріло, завжди 0
szExeFile               db MAX_PATH dup(?) ім'я модуля образу процесу
PROCESSENTRY32 ENDS
```

Обидві функції мають ANSI- і UNICODE-варіанти й по-різному трактують поле szExeFile структури PROCESSENTRY32.

Аналогічно вище розглянутому, для добування зі знімка потоків потрібної інформації використовується пара функцій Thread32First/Thread32Next, що заповнюють структуру

```
THREADENTRY32 STRUCT
dwSize                  DWORD ?  ; розмір цієї структури
cntUsage                DWORD ?  ; застаріло, завжди 0
th32ThreadID            DWORD ?  ; PID потоку
th32OwnerProcessID      DWORD ?  ; PID процесу потоку
tpBasePri               DWORD ?  ; базовий пріоритет
tpDeltaPri              DWORD ?  ; відносний пріоритет
dwFlags                 DWORD ?  ; застаріло, завжди 0
THREADENTRY32 ENDS
```

Для добування зі знімка модулів потрібної інформації використовується пара функцій Module32First/Module32Next, що заповнюють структуру

```
MODULEENTRY32 STRUCT
dwSize          DWORD ?  ; розмір цієї структури
th32ModuleID    DWORD ?  ; застаріло, завжди 1
th32ProcessID   DWORD ?  ; PID процесу модуля
GlblcntUsage    DWORD ?  ; зазвичай 0FFFFh
ProccntUsage    DWORD ?  ; зазвичай 0FFFFh
```

```
    modBaseAddr   DWORD ?  ; базова адреса модуля в контексті процесу
    modBaseSize   DWORD ?  ; розмір модуля в байтах
    hModule       DWORD ?  ; ідентифікатор модуля в контексті процесу
    szModule      db MAX_MODULE_NAME32+1 dup(?) ; ім'я модуля
    szExePath     db MAX_PATH dup(?)  ; шлях модуля
MODULEENTRY32 ENDS
```

Порядок виконання роботи

1. Завантажити із сервера кафедри архів з файлами поточної лабораторної роботи й розкрити його в будь-який каталог свого облікового запису.
2. Відкрити каталог проекту ProcessView.
3. Виконати запуск додатка ProcessView.exe.
4. Зазначити в звіті функції додатку ProcessView.exe.
5. Відкрити файл проекту ProcessView.rap.
6. Виконати трансляцію проекту. За результатами трансляції виправити синтаксичні помилки. Занести до звіту помилкові рядки, виправлені рядки та пояснення щодо виправлених помилок.
7. Виконати лінкування проекту. За результатами лінкування виправити помилки. Занести до звіту помилкові рядки, виправлені рядки та пояснення щодо виправлених помилок.
8. Виконати трансляцію і лінкування проекту та одержати програмний файл ProcessView.exe.
9. Переконатися, що робота отриманого в попередньому пункті додатка, відповідає його функціям, зазначеним раніше в п. 3. За необхідності виправити логічні помилки додатка і занести до звіту помилкові рядки, виправлені рядки та пояснення щодо виправлених помилок.
10. Модернізувати додаток ProcessView.exe таким чином, щоб він отримував у командному рядку ім'я процесу і у разі, якщо такий процес існує, виводив про нього інформацію, інакше – повідомлення про неіснуючий процес.

11. Модернізувати додаток ProcessView.exe таким чином, щоб в умовах виконання попереднього пункту він виводив інформацію і про всі модулі процесу із зазначенням базової адреси, розміру в пам’яті та імені програмного файла образу кожного модуля.

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 та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проекту модернізованого додатка з його детальними коментарями.

Контрольні питання

1. У Windows NT кожен потік виконується:
   а) у своєму власному адресному просторі;
   б) в адресному просторі ядра операційної системи;
   в) в адресному просторі процесу, якому він належить;
   г) в адресному просторі підсистеми оточення ОС.
2. Характерне значення величини кванту часу планування в сучасних ОС Windows має порядок:
   а) секунд;
   б) мілісекунд;
   в) мікросекунд;
   г) наносекунд.
3. Процес Windows NT з базовим пріоритетом шість може підвищити пріоритет своїх потоків до …

4. При зміні з восьми до десяти базового пріоритету процесу його потоки, які мали пріоритет дев’ять будуть мати пріоритет…
5. Комбінація класу відносного пріоритету потоку й класу базового пріоритету процесу дозволяє призначити значення шість як базовий пріоритет потоку
   а) 2-ма способами;
   б) 3-ма способами;
   в) 4-ма способами;
   г) 5-ма способами.
6. Потік може бути витиснутий…
   а) у будь-який момент;
   б) тільки в тому випадку, якщо є вільний процесор;
   в) тільки по закінченні свого кванта часу;
   г) якщо він був призупинений.
7. Пріоритет потоку звичайно підвищується…
   а) якщо даний потік знаходиться у стані очікування;
   б) по закінченні його кванту часу;
   в) якщо даний потік простоює протягом тривалого часу;
   г) по закінченні ним операції введення/виведення.
8. Для потоку А установлений відносний клас пріоритету THREAD_PRIORITY_ABOVE_NORMAL, а для потоку В – THREAD_PRIORITY_BELOW_NORMAL. Зазначити неправильне твердження
   а) пріоритет потоку А у будь-якому разі вищий за пріоритет потоку В;
   б) пріоритет потоку А може бути як нижчий, так і вищий за пріоритет потоку В;
   в) пріоритети потоків А і В збігаються;
   г) потоки ніколи не можуть мати такі відносні класи пріоритетів.

Література: *[1, Глава 3, с. 149 – 168; 2, Глава 4, с. 48 – 98; 5, Глава 4, с. 307 – 394; 6, Глава 6, с. 200 – 234; 7, Глава 3].*

Лабораторна робота № 5

Тема. Організація взаємодії поміж процесами за допомогою файлів, що відображені в пам’ять.

Мета: набуття практичних навичок організації взаємодії поміж процесами за допомогою файлів, що відображені в пам’ять та синхронізації обміну даними поміж процесами за допомогою подій, м’ютексів та семафорів.

Короткі теоретичні відомості

Практично всі механізми організації взаємодії поміж процесами, так чи інакше, базуються на спільному використанні об'єкта “розділ” (section object), що являє собою блок пам'яті, доступний двом і більше процесам, і є самим низькорівневиим базовим механізмом спільного використання даних.

Об'єкт “розділ” використовується при створенні об'єкта “проекція файла” (file-mapping object). Win API функції CreateFileMapping та MapViewOfFile(Ex) базуються на Native API функціях, які використовуються ядром для створення розділу і для відображення або проектування об'єкта “розділ” у віртуальний адресний простір заданого процесу і за своїми функціональними можливостями є по суті їх мало вкороченими аналогами. Тому файли, відображені на пам'ять, дозволяють реалізувати IPC через поділювану пам'ять при максимальній швидкодії з мінімумом витрат.

Спільне використання даних за допомогою об'єкта “розділ” виконується за наступним сценарієм. Один процес створює відображуваний на пам’ять файл, викликаючи функцію CreateFileMapping (див. лабораторну роботу № 3). Функція CreateFileMapping дозволяє, зокрема, створити відображення файла на пам'ять не пов'язане ні з яким дисковим файлом. Для цього як параметр ідентифікатора файла hFile у функцію потрібно передати значення 0FFFFFFFFh.

Об'єкт “розділ” є об'єктом ядра, і йому при створенні можна присвоїти ім'я. Таким чином, за допомогою CreateFileMapping можна створити

іменований об'єкт ядра «розділ» і використати цей ресурс разом з іншими процесами.

Потім, процес викликом функції MapViewOfFile відображає подання об'єкта “розділ” на свій адресний простір. Інший процес може відкрити цей самий відображуваний файл за допомогою функції

```
HANDLE OpenFileMapping(
  DWORD   dwDesiredAccess, // режим доступу
  BOOL    bInheritHandle,  // прапор спадкування
  LPCTSTR lpName);         // адреса імені відображення файла
```

Через параметр lpName цієї функції варто передати ім'я відображення, що відкриває. Ім'я повинне бути задане точно також, як при створенні відображення функцією CreateFileMapping.

Параметр dwDesiredAccess обумовлює необхідний режим доступу до відображення й указується точно також, як і для функції MapViewOfFile.

Параметр bInheritHandle визначає можливість спадкування ідентифікатора відображення. Якщо він дорівнює TRUE, породжені процеси можуть успадковувати ідентифікатор, якщо FALSE – то ні.

У випадку помилки OpenFileMapping повертає нульове значення, а у випадку успіху ідентифікатор відображення.

Відкривши відображення й маючи його ідентифікатор, інший процес може відобразити його викликом функції MapViewOfFile, але вже до свого адресного простору. У результаті ті самі фізичні сторінки пам'яті стають доступними в обох процесах, що дозволяє їм легко передавати через цю область великі порції даних, тому що будь-які зміни на цих сторінках в одному процесі відбиваються на їх подання в іншому процесі.

Порядок виконання роботи

1. Завантажити із сервера кафедри архів з файлами поточної лабораторної роботи й розкрити його в будь-який каталог свого облікового запису.
2. Відкрити каталог InterProcessMMF.

3. Виконати запуск додатків MMFServer.exe і WinObjEx.exe.
4. У меню Find додатка WinObjEx.exe вибрати Find Object… У вікні діалогу зазначити ім’я об’єкта – "$EventDataName", тип – Event. Знайти об’єкт, переглянути його властивості та занести до звіту тип і стан об’єкта.
5. Виконати п. 3 для об’єкта типу Section з іменем "$InterprocessCommunicationMMF$".
6. Перейти в консоль додатка MMFClient.exe (синього кольору), ввести декілька рядків з клавіатури, завершуючи їх натисканням Enter.
7. Перейти в консоль додатка MMFServer.exe й передивитися вміст вікна консолі.
8. Перейти в консоль додатка MMFClient.exe і натиснути Enter.
9. Зазначити у звіті функції додатків MMFServer.exe і MMFClient.exe.
10. Перейти у вікно WinObjEx.exe, натиснути F5 (Refresh) і повторити п. 3. та п. 4. Занести результати до звіту.
11. Відкрити каталог проекту MMFServer і файл проекту MMFServer.rap.
12. Модернізувати додаток MMFServer.exe таким чином, щоб він не сам створював процес клієнта, а чекав доки клієнт до нього приєднається.
13. Виконати запуск модернізованого додатка MMFServer.exe і чотирьох екземплярів додатка MMFClient.exe.
14. Повторити виконання п. 3 та п. 4 завдання.
15. Ввести по черзі з консолі всіх екземплярів додатків MMFClient.exe різні рядки символів.
16. Перейти в консоль додатка MMFServer.exe й передивитися вміст вікна консолі.
17. Перейти по черзі в консолі всіх екземплярів додатків MMFClient.exe і натиснути Enter.
18. Зазначити у звіті функції модернізованого додатка MMFServer.exe
19. Модернізувати додаток MMFServer.exe таким чином, щоб він був здатний підтримувати зв’язок одночасно не більше, ніж з трьома

клієнтами. Для контролю кількості клієнтів скористатися семафором, а для синхронізації доступу до файла, відображеного у пам’ять – м’ютексами.

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 та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проектів модернізованих додатків клієнта і сервера з їх детальними коментарями.

Контрольні питання

1. Зазначити неправильні твердження щодо функціонування м’ютексів Windows NT:

   а) структура, що забезпечує функціонування об'єкта ядра «м’ютекс» розміщується в пам'яті ядра;

   б) іменовані об'єкти ядра «м’ютекс» можна відкривати по імені з будь-якого потоку будь-якого процесу;

   в) для об'єктів ядра «м’ютекс» не передбачено можливості використання для синхронізації потоків одного процесу;

   г) потік, що створює об'єкт «м’ютекс», відразу може стати його власником і не буде виникати ніяких змагань;

   д) при використанні м’ютексів не виникає ніяких змагань потоків і тому не можливе їх взаємне блокування;

е) очікування протягом 0 мс звільнення м’ютекса виконується за допомогою функцій WaitForSingleObject або WaitForMultipleObjects з параметром dwMilliseconds, який дорівнює 0;

ж) функція WaitForMultipleObjects припускає можливість паралельного очікування звільнення 2-х м’ютексів;

и) ідентифікатори покинутих м’ютексів переходять у сигнальний стан.

2. Зазначити неправильні твердження щодо функціонування семафорів Windows NT:

а) семафори використовують для організації взаємовиключного послідовного доступу потоків до деякого ресурсу;

б) структура, що забезпечує функціонування об'єкта «семафор» розміщується в пам'яті ядра;

в) семафор може перебувати в сигнальному або несигнальному стані;

г) стан семафора визначається поточним значенням лічильника семафора;

д) якщо значення лічильника семафора більше за нуль, він перебуває в сигнальному стані;

е) якщо значення лічильника семафора стає нульовим, він переходить у несигнальний стан;

ж) кожна окрема операція очікування на Wait-функції може зменшити значення лічильника семафора тільки на 1;

и) виклик WaitForSingleObject для ідентифікатора семафора в сигнальному стані призводить до інкремента поточного значення лічильника семафора;

к) виклик WaitForSingleObject для ідентифікатора семафора в сигнальному стані призводить до декременту поточного значення лічильника семафора;

л) якщо значення лічильника семафора стає нульовим, він переходить у сигнальний стан;

м) коли поточний лічильник семафора стає більше нуля, стан семафора стає сигнальним, і робота потоку, що очікує на Wait-функції, відновлюється, при цьому значення поточного лічильника семафора декрементується;

н) ReleaseSemaphore може збільшити значення лічильника семафора відразу до будь-якого допустимого значення.

2. Зазначити правильні твердження щодо функціонування семафорів Windows

а) семафори використовують для організації взаємовиключного послідовного доступу потоків до деякого ресурсу;

б) звільнення семафора може бути виконано тільки тим потоком, який його захопив і тільки на одне значення лічильника;

в) очікування з використанням WaitForMultipleObjects дозволяє змінити значення лічильника семафора відразу до будь-якого допустимого значення;

г) кожний потік може захопити семафор тільки раз, і для семафорів не передбачено рекурсивного використання;

д) дізнатися поточне значення лічильника семафора можливо тільки змінивши його;

е) якщо значення лічильника семафора стає нульовим, він переходить у несигнальний стан;

ж) при створенні семафора не можливо уникнути змагань потоків за право володіння ним;

и) семафори дозволяють забезпечити паралельний доступ до ресурсу, але для обмеженої кількості потоків.

2. Зазначити неправильне твердження. Події, що скидаються вручну (manual-reset events)…

а) переводяться на несигнальний стан програмно, викликом спеціальної функції;

б) за відсутності потоків, що їх очікують залишаються в сигнальному стані доти, поки такий потік не з'явиться;

в) залишаються в сигнальному стані доти, поки який-небудь потік не викличе функцію скидання події;

г) при переході у сигнальний стан виводять зі стану очікування одночасно кілька потоків, що їх очікують.

2. Зазначити неправильне твердження. Події, що скидаються автоматично (auto-reset event)…

а) повертаються в несигнальний стан після звільнення тільки одного з потоків, що їх очікують;

б) за відсутності потоків, що їх очікують залишаються в сигнальному стані доти, поки такий потік не з'явиться;

в) за відсутності потоків, що їх очікують поводяться як семафори, з максимальним значення лічильника, який дорівнює 1;

г) створюються тільки в сигнальному стані.

Література: *[1, Глава 4, с. 189 – 220; 2, Глава 9, с. 207 – 245; 6, Глава 8, с. 271 – 293; 7, Глава 4].*

Лабораторна робота № 6

Тема. Організація взаємодії поміж процесами за допомогою іменованих каналів.

Мета: набуття практичних навичок організації взаємодії поміж процесами за допомогою іменованих каналів та синхронізації потоків у багатопоточних додатках за допомогою критичних секцій.

Короткі теоретичні відомості

Канали (pipes) – це найпростіший механізм, що надається операційною системою для реалізації IPC. Канали підтримуються не тільки Microsoft Windows NT, але й Unix-подібними ОС. Серед двох різновидів каналів у Windows NT більшою функціональністю володіють іменовані канали (named pipes), які дозволяють організувати передачу даних як між локальними процесами, так і між процесами, запущеними на різних робочих станціях у мережі та в різних режимах.

Через канал можна передавати дані тільки між двома процесами. Один із процесів створює канал, інший відкриває його за допомогою CreateFile і після цього обидва процеси можуть передавати дані через канал в один або в обидва боки, використовуючи для цього функції, призначені для роботи з файлами, такі як ReadFile й WriteFile. Процеси можуть виконувати над каналами як синхронні, так й асинхронні операції, аналогічно тому, як це можна робити з файлами.

Створення іменованого каналу (або його реалізації) виконується за допомогою функції

```
HANDLE CreateNamedPipe(
  LPCTSTR lpName,              // адреса рядка імені каналу
  DWORD   dwOpenMode,          // режим відкриття каналу
  DWORD   dwPipeMode,          // режим роботи каналу
  DWORD   nMaxInstances,       // максимальна кількість реалізацій каналу
  DWORD   nOutBufferSize,      // розмір вихідного буфера в байтах
  DWORD   nInBufferSize,       // розмір вхідного буфера в байтах
  DWORD   nDefaultTimeOut, // час очікування в мілісекундах
  LPSECURITY_ATTRIBUTES lpSecurityAttributes); // атрибути захисту
```

Через параметр lpName передається адреса рядка, що закривається нулем, з UNC-ім'ям каналу.

За допомогою параметра dwOpenMode можна вказати, чи буде даний канал використаний тільки для читання даних (PIPE_ACCESS_INBOUND),

тільки для запису (PIPE_ACCESS_OUTBOUND) або одночасно для читання й запису (PIPE_ACCESS_DUPLEX). Перераховані вище параметри повинні бути однакові для всіх реалізацій каналу. Додатково в параметрі dwOpenMode можна задати ще шість різних прапорів відповідальних за режим операцій введення/виведення й безпеку, які можуть бути різні для різних реалізацій каналу і які тут не розглядаються.

Параметр dwPipeMode, який визначає режим роботи каналу являє собою логічну суму констант із трьох різних груп. Константи в межах групи є взаємовиключними значеннями цього параметра.

PIPE_TYPE_BYTE або PIPE_TYPE_MESSAGE вказують, відповідно, чи будуть дані записуватися в канал як потік байтів, або як повідомлення. Даний параметр повинен бути однаковий для всіх реалізацій каналу. За замовчанням використовується PIPE_TYPE_BYTE.

PIPE_READMODE_BYTE або PIPE_READMODE_MESSAGE вказують, відповідно, чи будуть дані зчитуватися як потік байтів, або як повідомлення. Значення PIPE_READMODE_MESSAGE вимагає використання значення PIPE_TYPE_MESSAGE. За замовчанням використовується PIPE_READMODE_BYTE.

PIPE_WAIT або PIPE_NOWAIT – визначають, відповідно, буде або не буде канал працювати у блокую чому режимі. У блокую чому режимі потік, виконуючи операцію введення/виведення в каналі, переводиться в стан очікування до завершення операцій. У неблокуючому режимі, якщо операція не може бути виконана негайно, функція завершується з помилкою. За замовчанням використовується PIPE_WAIT.

Параметр nMaxInstances визначає максимальну кількість реалізацій, які можуть бути створені для каналу. Можна вказувати тут значення від 1 до PIPE_UNLIMITED_INSTANCES. В останньому випадку максимальна кількість реалізацій обмежується тільки наявністю вільних системних ресурсів. Якщо сервер працює з декількома клієнтськими процесами, то він може використати

для цього кілька реалізацій каналу, причому ті самі реалізації можуть застосовуватися по черзі.

Параметри nOutBufferSize й nInBufferSize визначають, відповідно, розмір буферів, використовуваних для запису в канал і читання з каналу. За необхідності система може використати буфери інших, порівняно із зазначеними розмірів.

Параметр nDefaultTimeOut визначає час очікування для реалізації каналу. Для всіх реалізацій необхідно вказувати однакове значення цього параметра.

Через параметр lpPipeAttributes передається адреса структури з атрибутами захисту для створюваного каналу.

У випадку успіху функція CreateNamedPipe повертає ідентифікатор створеної реалізації каналу, який можна використовувати в операціях читання й запису за допомогою функцій ReadFile й WriteFile. При помилці функція CreateNamedPipe повертає значення INVALID_HANDLE_VALUE.

Після того як серверний процес створив канал, він може перейти в режим з'єднання із клієнтським процесом. З'єднання з боку сервера виконується за допомогою функції ConnectNamedPipe.

```
BOOL ConnectNamedPipe(
  HANDLE hNamedPipe,                // ідентифікатор іменованого каналу
  LPOVERLAPPED lpOverlapped);   // адреса структури OVERLAPPED
```

Через параметр hNamedPipe серверний процес передає функції ідентифікатор каналу, отриманий від функції CreateNamedPipe.

Параметр lpOverlapped використовується тільки для організації асинхронного обміну даними через канал. Якщо використовуються тільки синхронні операції, як значення для цього параметра потрібно вказати NULL. Якщо параметр lpOverlapped установлений таким, що дорівнює NULL, то функція ConnectNamedPipe здійснює повернення відразу ж після встановлення з'єднання із клієнтом.

Для каналу, створеного в синхронному блокую чому режимі (з використанням константи PIPE_WAIT), функція ConnectNamedPipe переходить у стан очікування з'єднання із клієнтським процесом.

Якщо канал створений у синхронному неблокуючому режимі, функція ConnectNamedPipe негайно повертає керування з кодом TRUE, якщо клієнт був відключений від даної реалізації каналу й можливе підключення цього клієнта. У протилежному випадку повертається значення FALSE. Подальший аналіз необхідно виконувати за допомогою функції GetLastError. Ця функція може повернути значення ERROR_PIPE_LISTENING (якщо до сервера ще не підключений жоден клієнт), ERROR_PIPE_CONNECTED (якщо клієнт уже підключений) або ERROR_NO_DATA (якщо попередній клієнт відключився від сервера, але ще не завершив з'єднання).

Після повернення з функції ConnectNamedPipe сервер може виконувати читання запитів за допомогою функції ReadFile і запис відповідей за допомогою функції WriteFile.

Для з'єднання з каналом клієнтський процес використовує функцію CreateFile з параметром dwCreationDisposition, який дорівнює OPEN_EXISTING. Параметр lpFileName задається у вигляді UNC-імені згідно з правилами іменування каналів.

У випадку успішного завершення функція CreateFile повертає ідентифікатор реалізації каналу. При помилці повертається значення INVALID_HANDLE_VALUE.

Після успішного повернення з функції CreateFile клієнт може використовувати отриманий ідентифікатор для читання відповідей сервера за допомогою функції ReadFile і запису запитів за допомогою функції WriteFile.

Порядок виконання роботи

1. Завантажити із сервера кафедри архів з файлами поточної лабораторної роботи й розкрити його в будь-який каталог свого облікового запису.
2. Відкрити каталог InterProcessPipe.

3. Виконати запуск додатка PipeClient.exe. Зазначити у звіті результати виконання додатка PipeClient.exe.
4. Виконати запуск додатків PipeServer.exe і WinObjEx.exe. Зазначити у звіті результат виконання додатка PipeServer.exe.
5. У меню Extras додатка WinObjEx.exe вибрати Pipes…, знайти ім’я $NamedPipe$, виконати подвійний клік миші на імені, переглянути і занести до звіту його властивості.
6. Закрити додатки WinObjEx.exe і PipeServer.exe.
7. Виконати запуск додатків PipeServer.exe, PipeClient.exe і WinObjEx.exe. Знову виконати п. 4 і занести до звіту результати його виконання.
8. Ввести з консолі додатка PipeClient.exe команди «help», «date», «time», «exit». Зазначити у звіті функції додатків PipeServer.exe і PipeClient.exe.
9. Модернізувати додаток PipeServer.exe, додавши в нього підтримку команди «help». Отримавши цю команду, PipeServer.exe повинен повідомляти про перелік команд, які він підтримує.
10. Модернізувати додаток PipeServer.exe, зробивши його багатопоточним. Додаток повинен уміти підтримувати зв’язок через канал одночасно з трьома клієнтами, створивши для цього пул з трьох потоків. Первинний потік додатка, створивши пул потоків повинен відслідковувати час завершення сеансу зв’язку з клієнтом кожного потоку і створювати новий потік замість того, що вже завершується.
11. Проаналізувати модернізований додаток з метою виявлення ресурсів, які можуть використовуватися всіма потоками одночасно. Виконати за необхідності синхронізацію доступу потоків до спільних ресурсів за допомогою критичних секцій.

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 та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проекту модернізованих додатків клієнта та сервера з їх детальними коментарями.

Контрольні питання

1. Зазначити неправильне твердження. Іменовані канали (named pipes) …

   а) не висувають ніяких умов щодо привласнених їм імен, крім забезпечення їх унікальності;

   б) дозволяють скористатися вбудованими можливостями захисту Windows;

   в) підтримують два режими передачі даних – по байтовий (потоковий) і повідомлень;

   г) допускають багаторазове створення й одночасне існування декількох незалежних екземплярів каналу, що мають однакові імена й параметри.

2. Зазначити правильне твердження щодо API іменованих каналів Windows:

   а) установлення з’єднання виконується абсолютно однаковими засобами як з боку сервера, так і з боку клієнта;

   б) різні реалізації каналу можуть мати абсолютно різні значення будь-яких параметрів каналу за винятком імені;

   в) при створенні каналу з параметрами за замовчанням канал буде працювати в синхронному потоковому режимі;

   г) використання спеціальних функцій транзакцій іменованих каналів ніколи не призводить до втрат переданих даних.

3. Зазначити неправильні твердження щодо функціонування багатопотокових додатків ОС Windows NT:

   а) кожному потоку виділяється окремий стек з адресного простору процесу;

б) кожний потік має у своєму розпорядженні власний набір регістрів процесора;

в) адресові простори всіх потоків одного й того ж самого процесу ніколи не перетинаються;

г) операційна система виділяє кванти процесорного часу не процесу в цілому, а кожному з потоків процесу;

д) код потоків виконується тільки в контексті певного процесу;

е) потоки вимагають значно менше системних ресурсів, ніж процеси;

ж) потоки можуть виконувати один і той самий код і маніпулювати тими самими даними;

и) потоки одного процесу захищені від взаємного впливу один на одного;

к) усі потоки одного процесу працюють у єдиному адресному просторі;

л) оскільки таблиця ідентифікаторів об'єктів ядра створюється не в окремих потоках, а в процесах, потоки не можуть створювати об'єкти ядра і спільно їх використовувати;

м) можливе одночасне створення декількох потоків на основі однієї й тієї ж самої процедури, тобто з однаковим кодом.

4. Зазначити неправильне твердження щодо функціонування критичних секцій Windows NT. Критичні секції …

а) можна використовувати тільки для синхронізації потоків у межах одного процесу;

б) не припускають можливості рекурсивного захоплення одним і тим самим потоком;

в) дозволяють зробити так, щоб одночасно тільки один потік одержував доступ до певного ресурсу;

г) на відміну від об’єктів синхронізації ядра, працюють швидко, не знижуючи помітно продуктивність системи.

5. Причиною взаємного блокування потоків при синхронізації потоків за допомогою критичних секцій може бути:

   а) одночасне використання в процесі декількох різних екземплярів критичних секцій;

   б) використання в різних потоках процесу різної послідовності захоплення та звільнення критичних секцій;

   в) використання в потоках різних процесів різних екземплярів критичних секцій;

   г) спроба рекурсивного використання одного екземпляра критичної секції.

Література: *[1, Глава 6, с. 262 – 266, с. 252 – 354; 6, Глава 11, с. 363 – 391; 8, Глава 2].*

Лабораторна робота № 7

Тема. Організація взаємодії поміж батьківським та дочірнім процесом за допомогою анонімних каналів.

Мета: набуття практичних навичок організації взаємодії поміж процесами за допомогою не іменованих каналів та розробки програм з використанням стандартних елементів графічного інтерфейсу.

Короткі теоретичні відомості

Анонімні канали (anonymous pipes) Windows забезпечують односпрямовану побайтову передачу даних між процесами на локальному комп'ютері. За допомогою анонімного каналу не можна організувати IPC двох довільних процесів, не пов'язаних ні якими родинними відносинами. Причини таких обмежень стають зрозумілі при вивченні прототипу функції CreatePipe, що використовується для створення анонімного каналу:

```
BOOL CreatePipe(
  PHANDLE hReadPipe,
  PHANDLE hWritePipe,
  LPSECURITY_ATTRIBUTES lpPipeAttributes,
  DWORD nSize);
```

Канал може використовуватися або для запису в нього даних, або для читання. Тому при створенні каналу функція CreatePipe повертає два ідентифікатори, записуючи їх за адресами, заданими в параметрах hReadPipe й hWritePipe. Ідентифікатор, записаний за адресою hReadPipe, можна передавати як параметр функції ReadFile або ReadFileEx для виконання операції читання. Ідентифікатор, записаний за адресою hWritePipe, передається функції WriteFile або WriteFileEx для виконання операції запису.

Параметр nSize визначає розмір буфера для створюваного каналу. Якщо цей розмір зазначений як нуль, буде створений буфер з розміром, прийнятим за замовчанням. За необхідності система сама може змінити зазначений при створенні каналу розмір буфера.

Читання з використанням ідентифікатора читання каналу блокується, якщо канал порожній. У протилежному випадку в процесі читання буде прийнято стільки байтів, скільки є в каналі, аж до кількості, зазначеного при виклику функції ReadFile. Операція запису в заповнений канал, також буде блокована.

Через параметр lpPipeAttributes передається адреса структури з атрибутами захисту для створюваного каналу. Якщо вказати цей параметр як NULL, канал буде мати атрибути захисту, прийняті за замовчанням.

У випадку успіху функція CreatePipe повертає не нульове значення (TRUE), при помилці – FALSE.

Щоб канал можна було використати для IPC, повинен існувати ще один процес, і для цього процесу потрібен один з ідентифікаторів каналу. Тоді

виникає питання про те, як передати іншому процесу ідентифікатор читання або запису каналу.

Передачу ідентифікаторів створеного каналу іншому процесу можна виконати використовуючи механізм спадкування ідентифікаторів дочірніми процесами. Одним з найпоширеніших способів використання анонімних каналів є заміна ідентифікатора стандартного потоку введення або виведення дочірнього процесу, на один з ідентифікаторів каналу, що створив батьківський процес. Обмеженням такого механізму IPC є те, що дочірній процес повинен бути реалізований у вигляді консольного додатку ОС, бо тільки для консольних додатків існує поняття потоків введення/виведення.

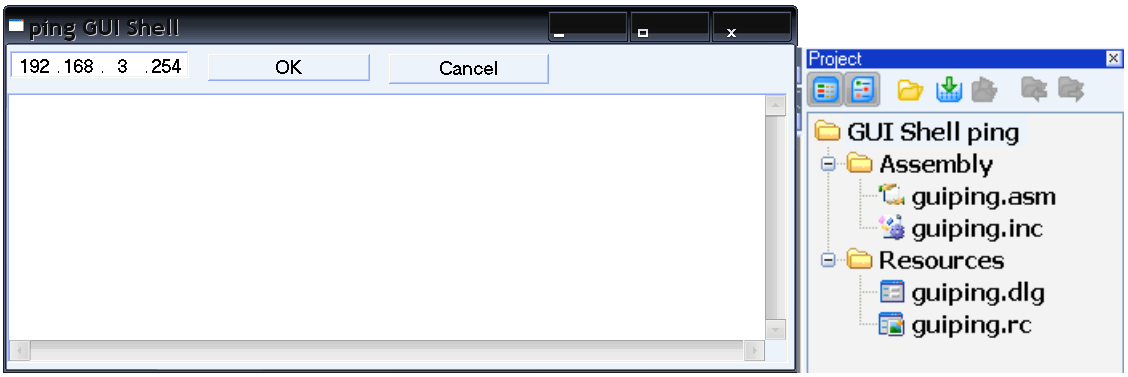
Щоб зробити ідентифікатор успадковуваним, батьківський процес повинен почати спеціальні дії, оскільки створювані ідентифікатори не стають такими за замовчанням. Створити успадковуваний ідентифікатор можна або шляхом використання структури SECURITY_ATTRIBUTES у момент створення об'єкта, або шляхом копіювання існуючого ідентифікатора за допомогою DuplicateHandle.

У структурі SECURITY_ATTRIBUTES наявний прапор bInheritHandle, значення якого повинно бути встановлене таким, що дорівнює TRUE. Також в екземплярі структури значенням sizeof SECURITY_ATTRIBUTES повинен бути ініціалізований елемент nLength.

При виклику функції CreateProcess повинен бути заданий прапор bInheritHandles, що визначає, чи буде дочірній процес успадковувати копії ідентифікаторів.

Порядок виконання роботи

1. Завантажити із сервера кафедри архів з файлами поточної лабораторної роботи й розкрити його в будь-який каталог свого облікового запису.
2. Відкрити каталог guiping, виконати запуск додатку guiping.exe і перевірити його роботу для адреси 192.168.3.254 (див. рисунок 1, а).



а) б)

Рисунок 1

3. Зазначити у звіті функції додатка guiping.exe.
4. Відкрити файл проекту guiping.rap і ознайомитися з вмістом усіх файлів проекту (див. рисунок 1, б).
5. Отримати допомогу з параметрів командного рядка утиліти ping.exe за допомогою ключа /?
6. Модернізувати додаток guiping.exe таким чином, щоб він підтримував ще параметри -n число, -a, -f та -j | -k командного рядка ping.exe:

а) модернізувати графічний інтерфейс користувача (файл guiping.dlg), додавши в нього групу (GroupBox) елементів RadioButton для зазначення параметрів -j | -k, групу елементів CheckBox для зазначення параметрів –a, -f, -n та будь-який з GUI-елементів для введення значення числа параметра –n;

б) внести за необхідності зміни у файл заголовка guiping.inc для підтримки нових елементів графічного інтерфейсу додатка;

в) модернізувати вихідний текст додатка guiping.asm, забезпечивши програмну підтримку нових елементів графічного інтерфейсу додатка.

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 та необхідні пояснення, як зазначено у відповідних пунктах робочого завдання.
4. Повний вихідний текст усіх файлів проекту модернізованого додатка з їх детальними коментарями.

Контрольні питання

1. Зазначити правильне твердження. Анонімні канали (anonymous pipes)…
   а) забезпечують однобічну потокову передачу даних між процесами на локальному комп'ютері;
   б) забезпечують двобічну передачу повідомлень між процесами на локальному комп'ютері;
   в) забезпечують однобічну потокову передачу даних між процесами на комп'ютерах локальної мережі;
   г) забезпечують двобічну передачу повідомлень між процесами на комп'ютерах локальної мережі.
2. Зазначити неправильне твердження. Анонімні канали (anonymous pipes) …
   а) дозволяють організувати IPC двох і більше довільних процесів, не пов'язаних ні якими родинними стосунками;
   б) можуть використовуватися або для запису в них даних, або для читання;
   в) надають ідентифікатори для читання/запису за допомогою основного API введення/виведення;
   г) підтримують виконання операцій введення/виведення в синхронному (блокованому) режимі.
3. Зазначити неправильне твердження, щодо анонімних каналів Windows NT:
   а) при недостатньому розмірі буфера анонімного каналу зазначеному при його створенні функцією CreatePipe операції читання/запису в каналі блокуються;
   б) за необхідності система сама може змінити розмір буфера зазначений при створенні анонімного каналу функцією CreatePipe;

в) читання з використанням ідентифікатора читання каналу анонімного каналу блокується, якщо канал порожній;

г) операція запису в заповнений канал блокується до його звільнення.

4. Зазначити правильне твердження. Щоб зробити ідентифікатор анонімного каналу успадкованим, батьківський процес:

а) створює канал із заданням відповідних параметрів структури SECURITY_ATTRIBUTES;

б) нічого не має робити, оскільки створювані ідентифікатори стають такими за замовчанням;

в) нічого не має робити, оскільки дочірній процес може виконати копіювання існуючого батьківського ідентифікатора за допомогою DuplicateHandle;

г) тільки повинен зазначити TRUE в елементі bInheritHandles структури STARTUPINFO при виклику функції CreateProcess.

Література: *[1, Глава 6, с. 280 – 284, с. 351 – 353; 4, с. 135 – 202; 6, Глава 11, с. 363 – 392; 7, Глава 4; 8, Глава 2].*

2 КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ СТУДЕНТАМИ

У 5-му семестрі студенти виконують 12 лабораторних робіт. Загальна кількість балів, яку отримують студенти за виконання та захист усіх лабораторних робіт складає 60 балів (5 балів за кожну лабораторну). У цій частини методичних вказівок наведено 7 лабораторних робіт. Отже за виконання та захист усіх лабораторних робіт цієї частини методичних вказівок студенти отримують 35 балів. Інші 5 лабораторних робіт наведені в наступній 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. Вильямс А. Системное программирование в Windows 2000 для профессионалов. − СПб.: Питер, 2001. − 624 с.; ил.
2. Рихтер Дж. Windows для профессионалов: создание эффективных Win32 приложений с учетом специфики 64-разрядной версии Windows/Пер. С англ. − 4-е изд. − СПб.: Питер; М.: Издательско-торговый дом "Русская Редакция", 2004. − 749 с.; ил.
3. Румянцев П. В. Работа с файлами в Win 32 API. − 2-е изд., − М.: Горячая Линия - Телеком, 2002. − 216 с.
4. Румянцев П.В. Азбука программирования в Win32 API. − 3-е изд., доп. − М.: Горячая Линия - Телеком, 2001. − 312 с.
5. Руссинович М. и Соломон Д. Внутреннее устройство Microsoft Windows Server 2003, Windows XP и Windows 2000. Мастер-класс. / Пер. с англ. − 4-е изд. − М.: Издательско-торговый дом "Русская редакция"; − СПб.: Питер; 2005. − 992 с.
6. Харт, Джонсон, М. Системное программирование в среде Windows. – 3-е издание: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 592 с.
7. Фролов А.В., Фролов Г.В. Программирование для Windows NT. − М.: Диалог-МИФИ, 1996. Том 26, − Часть 1. − 272 с.
8. Фролов А.В., Фролов Г.В. Программирование для Windows NT. − М.: Диалог-МИФИ, 1997. Том 27, − Часть 2. − 272 с.

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 І.

Укладачі: старш. викл. Ю.В. Зілінський,
старш. викл. О.М. Мотолига

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