УДК 004.451.2

Є. О. ЛОБОДА, Є. О. ТИМОФЕЙ

ВІДСТЕЖЕННЯ ПРОЦЕСІВ ІЗ РІЗНИМИ ОЗНАКАМИ У WINDOWS 7/8/10

Розроблено програмний модуль визначання переліку процесів, що виконуються в комп'ютері і надання інформації про їх поточний стан, ідентифікатори й пріоритети виконання активних процесів, перелік задіяних dll бібліотек. Метою науково-дослідної роботи було розробити програмний модуль сумісний з різними останніми версіями операційної системи Windows, який вперше забезпечує за допомогою діалогового вікна отримання розширеної інформації про всі діючи додатки та надає можливість позбавлення від тих з них, що вийшли з під контролю. Проведено тестування зробленої розробки додатку.

Ключові слова: модуль, процес, операційна система, контроль, ідентифікатор, пріоритет, тестування.

Разработан программный модуль определения перечня процессов, выполняемых в компьютере и отображения информации о их текущем состоянии, идентификаторы, приоритеты выполнения, перечень используемых dll библиотек. Целью научно-исследовательской работы было разработать программный модуль совместимый с различными последними версиями операционной системы Windows, который впервые обеспечивает с помощью диалогового окна получение расширенной информации о всех исполняемых приложениях и даёт возможность избавления от тех из них, которые вышли из под контроля. Выполнено тестирование работоспособности созданного приложения.

Ключевые слова: модуль, процесс, операционная система, контроль, идентификатор, приоритет тестирование.

A software module to determine the list of processes running on your computer and display information about their current status, IDs, perform the priorities list of used dll libraries. The aim of research was to develop a software module is compatible with a variety of the latest versions of the Windows operating system, which for the first time provides a dialog box to obtain extensive information on all executable applications and makes it possible to get rid of those that are out of control. Described sequentially all the steps of creating a conceptual project: carried out a detailed analysis of known attempts to accomplish the task; Described sequentially all the steps of creating a conceptual project: carried out a detailed analysis of the known attempts to perform the assigned tasks; We developed a method to obtain the optimal solution of the problem and justified project architecture with a given software functionality; logical design is made with the creation of the source code of software components and development of resources: physical design produced by running the executable. Functioning of the development has been tested on several versions of Windows. This testing fully confirmed the development operation, even when the process of getting rid of that came out under mortgaged for their control.

Keywords: module, a process, operating system, control, ID, priority, testing.

Вступ. Дуже часто користувачу комп'ютера важливо отримувати інформацію і спостерігати за станом вікон на екрані локального комп'ютера та дізнаватися детальну інформацію про них та їх зв'язок з іншими вікнами[1–11]. Наприклад: яке вікно є активне; яке знаходиться у фокусі; яке захопило управління мишею, особливо на великих екранах та якщо екранів декілька; як ця інформація виглядає з позиції різних інших вікон, тощо.

Постановка проблеми. Проведений пошук і аналіз існуючих програм про відстеження процесів з різними ознаками з отриманням інформації та спостеріганням за станом вікон на локальному комп'ютері та дізнанням детальної інформації їх зв'язку на жаль відсутні. Тому була поставлена задача написати новий програмний продукт.

Огляд існуючих засобів і технології рішення поставленого завдання. Особливість вирішення завдання утруднюється тім, що сучасні версії Windows не дозволяють прямий доступ проникнення у системні "таємниці" інших процесів. В сучасній операційній системі Windows застосована технологія, яка розділяє стан обміну між процесами системи для ізоляції їх один від одного. Можливості виконання завдання цієї наукової роботи можуть надати лише деякі, з великими обмеженнями системно орієнтовані API функції Windows.

Для отримання різноманітної інформації про систему в початкових версіях Windows надавався інтерфейс у вигляді лічильників продуктивності. Цей інтерфейс дозволяв отримати набір глибоко вкладених одна в одну структур. Для отримання будь-якої інформації потрібно прочитати з ключа реєстру HKEY_PERFORMANCE_DATA значення із спеціально сформованим ім'ям. У результаті повертається набір вкладених структур, багато з яких мають змінний розмір. З появою в Windows NT 4.0 лічильники продуктивності було надано у бібліотеці Performance Data Helper (PDH). Він став зручнішим для вимірювання даних, однак, ця бібліотека не входила в комплект постачання Windows NT 4.0, вона поширювалася лише у складі додатка Microsoft Platform SDK. Потім, з Windows 2000 вона була перенесена у PDH.DLL і стала присутньою за замовчуванням.

Система вимірювання лічильниками продуктивності в Windows реалізована за допомогою поняття об'єкта, для якого здійснюється підрахунок продуктивності. Кожен об'єкт може мати один або більше примірників, і для кожного об'єкта існує свій набір лічильників продуктивності. Завдання нашої розробки наполягає на отримання значень лічильників. Основна складність у використанні лічильників продуктивності полягає в тому, що назви об'єктів і лічильників продуктивності є локалізованими. Це означає, що, наприклад, на російській версії Windows необхідно використовувати зарезервовані константи «Процес» і «Ідентифікатор процесу» замість «Process» і «ID Process» для original версії Windows. Щоб отримати інформацію щодо процесів, як і для отримання локалізованих імен об'єктів і формування повного шляху до лічильника, необхідно писати допоміжну функцію, яка б здійснювала зворотне перетворення. Тож лічильники продуктивності

ISSN 2079-0023 (print)
ISSN 2410-2857 (online)

– це потужний і гнучкий механізм, але, він неочевидний і незручний.

На апаратному (внутрішньому) рівні інформація від пристроїв (клавіатури, миші, …) потрапляє в систему і пересилається відповідним віконним процедурам. Створюючи модель технології введення даних, Microsoft прагнула, щоб жодний потік не міг порушити роботу інших потоків. Загальна схема моделі апаратного введення в Windows системі зображена нижче на рис. 1 для приклада використанні чотирьох вікон (A1, B1, B2, C1). При запуску система створює собі особливий потік необробленого вводу RIT (Raw Input Thread) і системну чергу апаратного введення SHIQ (System Hardware Input Queue). RIT і SHIQ – це фундамент, на якому побудована вся модель апаратного введення Windows.

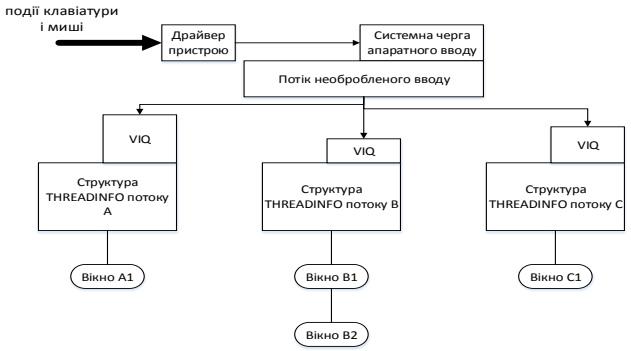

Рис. 1 – Модель апаратного вводу

Зазвичай RIT бездіяльна, чекаючи появи якогось елемента в SHIQ коли користувач натискає і відпускає клавішу на клавіатурі або кнопку миші, або переміщує мишу, відповідний драйвер пристрою додає апаратну подію у SHIQ. Тоді RIT пробуджується, витягує цей елемент із SHIQ, перетворює його в повідомлення (WM_KEY*, WM_?BUTTON*, WM_MOUSEMOVE, …) і ставить в кінець черги віртуального введення VIQ (Virtualized Input Queue) потрібного потоку. Далі RIT повертається в початок циклу і чекає появи наступного елемента в SHIQ. RIT ніколи не перестає реагувати на події апаратного введення – весь його код написаний самою фірмою Microsoft і дуже ретельно протестований.

Введення з клавіатури направляється потоком необробленого вводу (RIT) у чергу віртуального введення якогось потоку, але тільки не у вікно RIT події від клавіатури в чергу потоку безвідносно конкретного вікна. Коли потік викликає GetMessage(), подія від клавіатури витягується з черги і перенаправляється вікну (створеному потоком), на якому в даний момент зосередили фокус введення. Щоб направити клавіатурний ввід в інше вікно, потрібно вказати, в чергу якого потоку RIT необхідно розмітити події від клавіатури, а також «повідомити» змінним стану вводу потоку, яке вікно буде перебувати у фокусі. Вікно, що втрачає фокус, прибирає використовуваний для позначення фокусу прямокутник або гасить курсор введення, а вікно, яке отримує фокус, малює такий прямокутник або показує курсор введення. RIT направляє користувача введення з клавіатури в чергу віртуального введення тільки одного з потоків одноразово.

Механізм роботи з потоками процесу ідентичний механізму, описаному для модулів, тільки викликаються інші API Windows.

Доступ до пам’яті процесу має великі труднощі. Відомо, що в Win32 отримати доступ до пам’яті одного процесу з іншого офіційно неможливо. Проте в THL є функція ProcessMemory(), що дозволяє отримати копію блоку пам’яті іншого процесу. Для дослідження динамічної пам’яті процесу виділяються області у віртуальній пам’яті, яка створює відповідний об’єкт (heap) в адресному просторі процесу. З допомогою API можливе виділення пам’яті у середині цього.

Вікна, створені в різних потоках зазвичай обробляють введення даних незалежно один від одного. Тобто вони мають свої власні стани введення даних (фокус; активність; захоплення мишею, чи ні; стан клавіші; стан черги і так далі). Ці дані синхронізовані з обробкою введення даних інших потоків.

Для підвищення ізоляції потоків, один від одного, система підтримує додаткову концепцію – локальний стан введення LIS (Local Input State). Кожен потік володіє власним станом вводу, відомості про який зберігаються в структурі THREADINFO. В інформацію про цей стан включаються дані про чергу віртуального введення потоку і група змінних, яка містить керуючу інформацію про стан введення.

Відносно клавіатури підтримуються наступні відомості:

1) яке вікно перебуває у фокусі клавіатури;
2) яке вікно активно в даний момент;
3) які клавіші натиснуто;
4) стан курсору введення.

Відносно миші підтримуються наступні відомості:

1) яким вікном захоплена миша;
2) яка форма курсору миші;
3) чи бачимо цей курсор.

Так як у кожного потоку свій набір змінних стану вводу, то й представлення про вікно, що знаходиться у фокусі, про вікно, що захопило мишу, і т. п. у них теж суто свої. З точки зору потоку, клавіатурний фокус або є у одного з вікон, або його немає в жодного вікна у всій системі. Те ж саме відноситься і до миші або вона захоплена одним з вікон, або не захоплена ніким. В додаток до цього, є API функція AttachThreadInput(), яка надає інформацію про спільне використання однієї черги різними потоками та може змінити локальний стан введення і черги віртуального введення.

Операційна система неявно з’єднує черги віртуального введення двох потоків, якщо якийсь із них встановлює пастку реєстрації (journal record hook) або пастку відтворення (journal playback hook). Коли пастка знімається, система відновлює схему організації черги введення, що існувала до установки

пастки. Установкою пастки реєстрації потік повідомляє, що хоче отримувати повідомлення про всіх апаратні події, що викликаються користувачем. Потік зазвичай зберігає або реєструє цю інформацію в файлі. Так як користувальницьке введення повинне бути зареєстроване в тому порядку, в якому воно відбувалося, всі потоки в системі починають розділяти одну чергу віртуального введення для синхронізації обробки введення. Відмовостійкість моделі введення досягається завдяки тому, що у кожного потоку є власні змінні локального стану введення, а підключення потоку до RIT і відключення від нього відбувається по мірі необхідності.

З точки зору потоку, клавіатурний фокус або є у одного з вікон, або його немає у жодного вікна, у всій системі. Те ж саме відноситься і до миші або вона захоплена одним з вікон, або не захоплена ніким.

Windows дозволяє додаткам створювати і управляти призначеним для користувача інтерфейсом. Додатки визначають загальну поведінку і зовнішній вигляд їх вікон, створюючи класи вікон і відповідні процедури вікна. Клас вікна ідентифікує замовчуванням такі характеристики, як: чи обробляє вікно клік миші або має чи ні воно меню. Відповідна процедура вікна містить код, який визначає поведінку вікна, оброблює завдання і введенні користувачем дані. Додатки отримують зв'язок з мишею і клавіатурою у вигляді повідомлень. Система перетворює рухи миші, кліки миші і натиснення клавіш на вхідні повідомлення і поміщує ці повідомлення в чергу повідомлень для додатка. Система автоматично забезпечує черги повідомлень для кожного додатку. Додаток використовує повідомлення функцій для вилучення повідомлень з черги і відправляє їх у відповідну процедуру вікна для обробки.

Постановка завдання. За результатами наведеного вище огляду існуючих засобів отримання інформації про вікна було прийнято рішення: розробите програмне забезпечення з діалоговим вікном з виводом на ньому щосекундну інформацію про слідуючи стани вікон: активне вікно, що знаходиться у фокусі та вікна, які захопили управління мишею. Також програмний продукт повинен мати змогу спостерігати за локальним станом введення тільки в одному потоці та коректно повідомляти інформацію про вікна незалежно від того, який потік створив це чи інше вікно. Необхідно забезпечувати відсутність тупикових ситуацій у ході своєї роботи. При виникненні помилки – коректно завершувати роботу, не впливаючи на роботу і стан операційної системи Windows. Розробка повинна коректно функціонувати під керуванням всіх різних версій Windows, від Windows XP до Windows 10.

Було визначено, що для розробки діючого проекту найбільш доцільним є використання в ньому API функції Windows AttachThreadInput() через кращу стабільність та підтримку цього методу більшою кількістю останніх версій операційної системи Windows. Ця функція створена компанією Microsoft, отже програми, що будуть її використовувати, зненацька не втратять свою працездатність [3]. Дана функція дозволяє підключати або вимикати механізми обробки вводу даних одного з потоків до механізму іншого потоку. Переваги застосування цього методу в тому, що є можливість керування інформацією про стан вікон:

1) один потік може підключати свою обробку вводу до іншого потоку;
2) потоки можуть спільно використовувати свої статуси введення даних;
3) потік може забезпечити установку фокусу клавіатури в вікні іншого потоку.

Для застосування такої програми не потрібні спеціалізовані сервери, багатопроцесорні системи та мережі.

Обґрунтування архітектури створюваного проекту. Для відповідності до сучасних вимог розробки було прийнято рішення:

- мати діалоговий інтерфейс взаємодії з користувачем;
- виводити детальну інформацію про стан вікон в реальному часі з обновленням щосекунди;
- спостерігати за локальним станом введення тільки одного визначеного потоку;
- мати сумісність з усіма останніми версіями операційної системи Windows.

Керування програмою користувач виконує кою. Інформація відображається в елементах діалогового вікна. Для реалізації такої програми не потрібні спеціалізовані сервери, багатопроцесорні системи та мережі.

Тестування створеного проекту. Після запуску виконуючого файлу зробленого проекту побачимо діалогове вікно з різними елементами графічного інтерфейсу користувача (рис. 2)

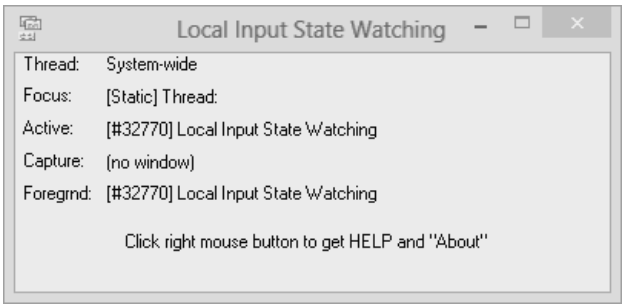

Рис. 2 – Початковий вид діалогового вікна

Після запуску програма відстежує активізацію вікон в усіх частинах системи, на всіх екранах комп'ютера. На це вказує надпис System-wide у верхній частині діалогового вікна. Але розробка Local Input State Watching може спостерігати за локальним станом введення тільки в одному потоці. Якщо вибрати цей режим, програма буде повідомляти, що «думає» потік про свій стан введення. Щоби програма контролювала стан введення тільки в одному потоці, треба клацнути лівою кнопкою миші, та, не відпускаючи кнопку, перемістити курсор миші у вікно, яке створено іншим потоком, після цього

відпустити кнопку. Наприклад, запустимо «Блокнот» та виберемо його вікно в Local Input State Watching. Отримаємо результат приведений на рис. 3.

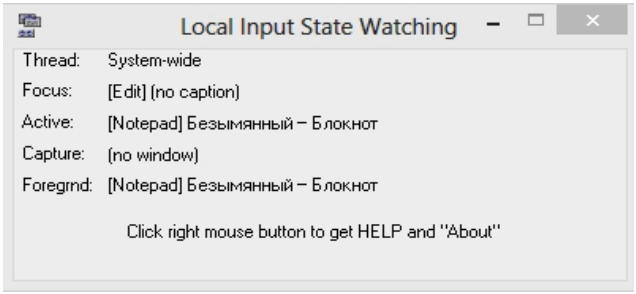

Рис 3 – Відслідковування стану вікна та фокусу вводу

Розглянемо іншу функціональність – запустимо стандартний «Калькулятор» та виберемо його вікно в Local Input State Watching то побачимо таку інформацію, що відображена на рис. 4.

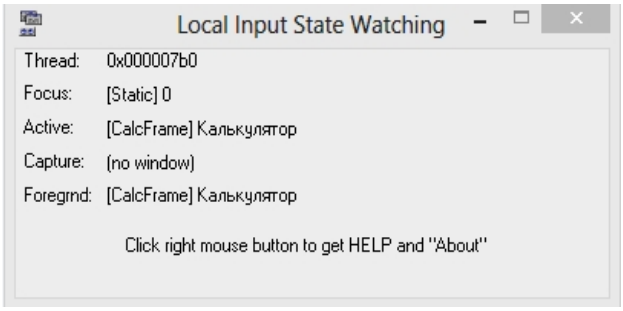

Рис. 4 – Вибір спостереження за локальним станомвведення тільки одного потоку

Ідентифікатор потоку Калькулятора отримав значення 0x000007b0. В цьому випадку програма буде налаштована на спостереження за локальним станом введення тільки цього потоку. Але якщо клацнути будь-яку кнопку або зайти в меню, то Local Input State Watching негайно відреагує на це, повідомивши відповідні зміні фокусу.

Однак, якщо тепер активізувати вікно, створене іншим додатком, наприклад «Мій комп'ютер», вікно Local Input State Watching буде виглядати так, як це зображено на рис. 5.

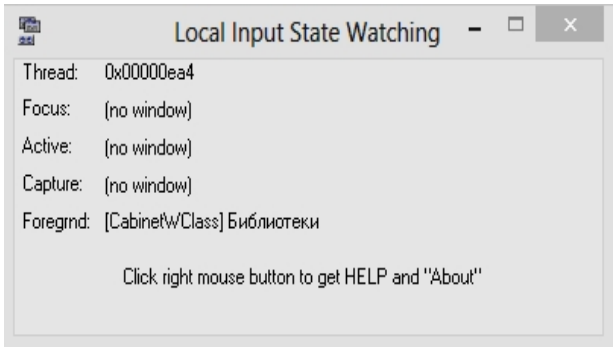

Рис. 5 – Активізація вікна іншого додатка при спостереженні за першим

Як можна побачити з рис. 5, «Калькулятор» вважає що немає не одного з вікон, які б знаходились у фокусі, були б активними, чи захопили мишу.

Висновки. Тестування розробки «Відстеження процесів з різними ознаками» було успішно проведено на кількох останніх версіях Windows, та їх підмножинах: Windows Mobile, Windows CE, Microsoft Silverlight, NET Framework і NET Compact Framework. Тобто можна казати про досягнення наданої сумісності розробки, що була одним з критеріїв вибору даного методу відстеження процесів. Ніяких збоїв у роботі розробки не виникало.

Список літератури:

1. *Руссинович М.* Внутреннее устройство Microsoft Windows / *М. Руссинович, Д. Соломон*. – М. : Русская редакция; СПб. : Питер, 2008. – 992 с.
2. *Рихтер Д.* WinRT: программирование на C# для профессионалов = Windows Runtime via C# / *Джеффри Рихтер, Мартен ван де Боспурт*. – М. : Вильямс, 2014. – 368 с.
3. *Джонсон М. Х.* Системное программирование в среде Windows / *М. Х. Джонсон*. – М. : Вильямс, 2005. – 592 с
4. *Deitel H. M.* C++ for Programmers, 2nd Edition Publisher / *Harvey M. Deitel*. – Prentice Hall, 2013. – 848 с.
5. *Machado M.* Customizing the Microsoft NET Framework Common Language Runtime / *Manel Machado*. – 2008. – 896 p.
6. *Deshpande M.* Item-based top-N recommendation algorithms / *M. Deshpande, G. Karypis* // ACM Transactions on Information Systems. – 2011. – Vol. 29, no. 1. – P. 143–177.
7. *Nikovski D.* Induction of compact decision trees for personalized recommendation / *D. Nikovski, V. Kulev* // Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006). Dijon, France, April 2006. – Vol. 1. – Dijon, 2006. – P. 575–581.
8. *Su X.* A survey of collaborative filtering techniques. – Advances in Artificial Intelligence/ *Xiaoyuan Su, Taghi M. Khoshgoftaar*. – New York : Hindawi Publishing Corporation, 2009. – P. 1–19. – doi: 10.1155/2009/421425
9. *Linden G.* Item-to-Item Collaborative Filtering / *G. Linden, B. Smith, J. York* // IEEE Internet Computing. – Los Alamitos, CA USA. – 2003. – P. 76–80.
10. *Рихтер Д.* Создание эффективных WIN32-приложений с учётом специфики 64-разрядной версии Windows. – *Джеффри Рихтер*. – СПб. : Питер «Русская редакция», 2011. – 752 с.
11. *Krizhevsky A.* ImageNet classification with deep convolutional neural networks / *Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton* // Advances in Neural Information Processing Systems 25 (NIPS 2012). Vol. 25. – Lake Tahoe, Nevada, 2012. – P. 1097-1105.

Bibliography (transliterated)

1. Russinovich M., Solomon D. *Vnutrennee ustroystvo Microsoft Windows* [Windows Internals]. Moscow, Russkaja Redakcija Publ. and Saint Petersburg, Piter Publ., 2008. – 992 p.
2. Jeffrey Richter, Marten van de Bospoort. *WinRT: prohrammyrovanye na C# dlya professyonalov* [Windows Runtime via C#]. Moscow, Vil'jams Publ., 2014. 368 p.
3. Johnson M. Hart. *Systemnoe prohrammyrovanye v srede Windows* [Windows System Programming]. Moscow, Vil'jams Publ., 2005. 592 p.
4. Deitel Paul, Deitel Harvey. *C++11 for Programmers (2nd Edition)*. Pearson Education Inc. Publ., 2013. 848 p.
5. Manel Machado Customizing the Microsoft .NET Framework Common Language Runtime 2008. 896 p.
6. Deshpande M., Karypis G. Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems*. 2011, vol. 29, no. 1, pp. 143–177.
7. Nikovski D., Kulev V. Induction of compact decision trees for personalized recommendation. *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006). Vol. 1*. Dijon, France, April 2006, pp. 575–581.
8. Su Xiaoyuan, Khoshgoftaar Taghi M. A survey of collaborative filtering techniques. Advances in Artificial Intelligence. New York, Hindawi Publishing Corporation, 2009, pp. 1–19. doi: 10.1155/2009/421425
9. Linden G., Smith B., York J. Item-to-Item Collaborative Filtering. *IEEE Internet Computing*. Los Alamitos, CA USA, 2003, p. 76–80

10. Richter J. Sozdanie jeffektivnyh WIN32-prilozhenij s uchjotom specifiki 64-razrjadnoj versii Windows [Programming Applications for Microsoft WINDOWS]. – Saint Petersburg, Piter Publ. and Moscow, Russkaja Redakcija Publ., 2011. – 752 p.

11. Krizhevsky A., Sutskever I., Hinton G. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25 (NIPS 2012). Vol. 25*. Lake Tahoe, Nevada, 2012, pp. 1097–1105.

Надійшла (received) 12.05.2016

Бібліографічні описи / Библиографические описания / Bibliographic descriptions

Відстеження процесів із різними ознаками у Windows 7/8/10 / Є. О. Лобода, Є. О. Тимофей // Вісник НТУ «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – Харків : НТУ «ХПІ», 2016. – № 28 (1250). – С. 65–69. – Бібліогр.: 11 назв. – ISSN 2079-0023.

Отслеживание процессов с различными признаками в Windows 7/8/10 / Е. А. Лобода, Е. О. Тимофей // Вісник НТУ «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – Харків : НТУ «ХПІ», 2016. – № 28 (1250). – С. 65–69. – Библиогр.: 11 назв. – ISSN 2079-0023.

Tracking processes with different characteristics in Windows 7/8/10 / E. A. Loboda, E. O. Timofey // Bulletin of NTU "KhPI". Series: System analysis, control and information technology. – Kharkov : NTU "KhPI", 2016. – No. 28 (1250). – P. 65–69. – Bibliogr.: 11. – ISSN 2079-0023.

Відомості про авторів / Сведения об авторах / About the Authors

Лобода Євген Олександрович – кандидат технічних наук, доцент, Національний технічний університет «Харківський політехнічний інститут», професор кафедри обчислювальної техніки та програмування; тел.: +38(057) 336-41-68; e-mail: loboda.eugene@gmail.com

Лобода Евгений Александрович – кандидат технических наук, доцент, Национальный технический университет «Харьковский политехнический институт», профессор кафедры вычислительной техники и программирования; г. Харьков, Украина; тел.: +38(057) 336-41-68; e-mail: loboda.eugene@gmail.com

Loboda Eugene Alexandrovich – Candidate of Technical Sciences (Ph. D.), Docent, National Technical University "Kharkiv Polytechnic Institute", Professor at the Department of Computer Engineering and Programing; tel.: +38(057) 336-41-68; e-mail: loboda.eugene@gmail.com

Тимофей Євген Олегович – Національний технічний університет «Харківський політехнічний інститут»; студент факультету комп'ютерних та інформаційних технологій; тел.: +38(098) 542-53-73).

Тимофей Евгений Олегович – Национальный технический университет «Харьковский политехнический институт»; студент факультета компьютерных и информационных технологий; тел.: +38(098) 542-53-73).

Timofey Eugene Olegovich – National Technical University "Kharkiv Polytechnic Institute", student of Faculty of Computer and Information Technology, tel.: +38(098) 542-53-73.