УДК 004.42 doi:10.20998/2413-4295.2022.04.05

РОЗРОБКА ТЕКСТОВОГО РЕДАКТОРА ДЛЯ ОПЕРАЦІЙНОЇ СИСТЕМИ UNIX

В. О. МЄТЄЛЬОВ*, О. М. МАРУСЕНКО, О. Р. БАСКАКОВ

кафедра комп'ютерного моделювання процесів та систем, НТУ «ХПІ», Харків, УКРАЇНА
**e-mail: volodymyr.mietielov@khpi.edu.ua*

АНОТАЦІЯ *Розглядається задача розробки та реалізації програмного додатку для створення, редагування, виведення на екран, а також збереження у вигляді файлів різного роду форматів, які у свою чергу будуть використовуватися у програмуванні на таких мовах як C, C++, Python, та системної мови Linux – Bash. Розроблений програмний додаток призначений для надання можливості користувачу взаємодії з файлами різного розширення, редагування та збереження змін, робота зі змістом файлів для подальшої розробки програмного коду на таких мовах, як C, C++, Python. Програмний додаток можливо використовувати як консольний додаток, так і з використанням графічного інтерфейсу на операційних системах UNIX та MS Windows. При розробці використовувалась операційна система UNIX, а саме дистрибутив Linux – Ubuntu, це надало змогу для використання створеного текстового редактора, на таких операційних системах як Ubuntu, Linux Mint, Kali Linux, Raspberry Pi Ubuntu. Також був використаний вільний кросплатформений мультимедійний пакет бібліотек – SFML, що надав змогу для використання текстового редактора на платформі MS Windows. Вихідний код бібліотеки SFML надається під ліцензійною угодою для розповсюдження вільного програмного забезпечення ZLIB/PNG License. Під час розробки визначено та реалізовано метод обробки вхідної інформації, а також збереження у файл окремого формату. Були використані наступні мови програмування: C, C++11, розроблена зручна архітектура програмного додатку, яка дозволяє з легкістю підтримувати та удосконалювати програму у майбутньому. Використання віртуальної машини для проекту було невід'ємною складовою. Через використання віртуальної машини, багато ресурсів комп'ютера утилізується, наприклад, оперативна пам'ять, кількість ядер, сховище даних та дискретне прискорення робочого стола у бажаної операційної системи. Реалізовано дружній графічний інтерфейс для взаємодії із користувачем.*
Ключові слова: *операційні системи; текстовий редактор; UNIX; Windows; бібліотека SFML; Bash; C++*

DEVELOPMENT OF A TEXT EDITOR FOR UNIX OPERATING SYSTEM

V. MIETIELOV*, O. MARUSENKO, O. BASKAKOV[1]

Department of Computer Modeling of Processes and Systems, NTU "KhPI," Kharkiv, UKRAINE

ABSTRACT *The task of developing and implementing a software application for creating, editing, displaying on the screen, as well as saving in the form of files of various formats, which in turn will be used in programming in languages such as C, C++, Python, and the Linux system language – Bash, is being considered. The developed software application is designed to enable the user to interact with files of various extensions, edit and save changes, work with the content of files for further development of software code in languages such as C, C++, Python. The software application can be used both as a console application and with the use of a graphical interface on UNIX and MS Windows operating systems. During development, the UNIX operating system was used, namely the Linux – Ubuntu distribution, which made it possible to use the created text editor on such operating systems as Ubuntu, Linux Mint, Kali Linux, Raspberry Pi Ubuntu. A free cross–platform multimedia package of libraries – SFML was also used, which made it possible to use a text editor on the MS Windows platform. The source code of the SFML library is provided under the ZLIB/PNG Free Software Distribution License. During development, a method of processing input information, as well as saving it in a file of a separate format, was defined and implemented. The following programming languages were used: C, C++11, a convenient software application architecture was developed, which allows you to easily maintain and improve the program in the future. The use of a virtual machine was an integral part of the project. By using a virtual machine, many computer resources are utilized, such as: RAM, number of cores, data storage and discrete desktop acceleration in the desired operating system. A friendly graphical interface for user interaction has been implemented.*
Keywords*: operating systems; text editor; UNIX; Windows; SFML library; Bash; C++*

Вступ

Текстові редактори [1] різняться за своїм призначенням, складністю, функціональними можливостями. Зручні та компактні у розмірі програмні редактори, мають високу популярність серед системних менеджерів, адміністраторів і програмістів.

Швидкість та змога редактора працювати зі специфікаціями (кожна операційна система має свої допоміжні механізми для роботи з даними та файлами) операційної системи є майже головним фактором у виборі програмного продукту для менеджменту файлів та їх вмісту.

Використання такої мови як C++ надає змогу для реалізації як швидкого, так і чітко структурованого програмного рішення. Мови C та C++ мають особливі механізми для роботи з файловою системою операційних систем як Microsoft Windows, так і систем Linux, а саме управління щонайменше кожним бітом (1 bit), та виділення чітко необхідної купи сторінок, віртуальної оперативної

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

пам'яті (як правило використовуються сторінки розміром у 4 кілобайта (4 Kb) для подальшого контролю створення файлів.

У роботі програмний додаток створювався на базі операційної системи Linux, а саме – дистрибутиву Linux–Ubuntu [2]. Це надало змогу для використання створеного текстового редактора на таких ОС як Ubuntu, Linux Mint, Kali Linux, Raspberry Pi Ubuntu. Також був використаний пакет бібліотек – SFML, що надав змогу для використання програми на платформі MS Windows. Вихідний код бібліотеки SFML надається під ліцензією ZLIB/PNG License.

Для початку роботи у програмі необхідно її викликати, це можливо зробити як за допомогою ярлика (icon, link) самого редактора, так і за допомогою терміналу Linux. Саме використання редактора через термінал надає змогу для утилізації такого функціоналу як транспортування виводу системних додатків (pipe, " | " – синтаксис написання у терміналі). Такого роду функціонал дуже популярний серед системних програмістів складних систем та автоматизації виконання завдань.

Мета роботи

Мета цієї роботи – розробка та реалізація програмного додатку для створення, редагування, виведення на екран, а також збереження у вигляді файлів різного роду форматів, які у свою чергу будуть використовуватися у програмуванні на таких мовах як C, C++, Python, та системної мови Linux-Bash.

Розроблений текстовий редактор сумісний з операційною системою на базі UNIX–Linux Ubuntu, Linux Mint, Kali Linux. Також, для сумісності програмного рішення з операційними системами Microsoft Windows використана мультимедійна бібліотека SFML, яка у свою чергу є простим платформним шаром над Win32 API, Linux Core.

Розроблена програма працює як консольний додаток, так і як самостійний віконний модуль із самостійним інтерфейсом.

Виклад основного матеріалу

У роботі використана операційна система MS Windows, як головна система для написання редактора на UNIX системи. Зручний механізм використання віртуальної машини із дистрибутивом Linux Ubuntu, дозволяє тестувати роботу текстового редактора, як на MS Windows, так і на Linux системах.

Вибір дистрибутиву Linux Ubuntu має логічне пояснення, а саме те, що ця ОС гарно підтримується розробниками та має більше користувачів цієї ОС, ніж інших дистрибутивів. Якщо програмний додаток працює на Linux Ubuntu, то програма буде працювати і на таких системах як Kali Linux, Linux Mint, RaspberryPi Ubuntu.

Використання компілятора GCC/C++ [3,4] є рішенням схожості його до системного компілятору на платформі MS Windows. Тоді як GCC та MS VCVARSALL схожі, редактор буде компілюватися майже однаково для двох операційних систем.

SFML – це не є набір готових рішень для побудови програм [5,6]. Ця бібліотека має одне призначення, а саме, бути платформним шаром між операційною системою (ОС) і програмістом, який не бажає витрачати час, або не знає про системне програмування. Функціонал даної бібліотеки допомагає уникати складних конструкцій при програмуванні рішення.

Мова C/C++ була вибрана не випадково, тому що саме на цій мові написані ОС MS Windows та Linux. Через це бібліотека SFML також розроблена на мові C++. Також використання гарного, вбудованого функціоналу мови C, а саме робота з даними, побітовий здвиг, якщо необхідно, маніпуляція з файлами будь-якого формату – має велике значення для проекту текстового редактора.

Опис роботи текстового редактора має декілька етапів. Розділи програмного коду, з яких буде складатися опис: головної функції main; робота редактора з консолі; робота вікна у ОС; навігація у додатку.

Стисла робота головної функції main представлена нижче:

– робота програми розпочинається зі зчитування аргументів із терміналу за допомогою атрибутів;

– для відображення робочого вікна надаємо наступні параметри: розмір вікна, колір та назва вікна, як заголовок;

– створення екземпляра класу TextDocument для подальшої праці із редагуванням файла;

– створення екземпляра класу EditorContent – класу навігації;

– ініціюємо функціонал класу InputController – клас прийому контролю для додатка;

– виконується запис у файл.

Основні функції роботи текстового редактора у консолі не відрізняються від роботи програми з графічним інтерфейсом. За окремістю того, що нам не треба обробляти роботу GUI, і за нас це робить операційна система.

Весь функціонал для роботи як консольного додатку (рис. 1) надає операційна система UNIX [7–11].

Якщо розглядати роботу вікна у операційній системі Linux Ubuntu та MS Windows, вони будуть дуже сильно відрізнятися, як на програмному рівні самого API, так і на рівні роботи системного коду. Наприклад, для програмування інтерфейса в ОС Linux як правило використовується X11 Server API, тому що Linux має зовсім інші системні «сигнали», та документація для API не надасть зрозумілості, якщо описувати роботу текстового редактора на ОС Linux. Для абстрактності опис роботи надано на рівні SFML та системних викликів із під ОС MS Windows, яка має більш прозорий API і документацію.

Рис. 1 – Редагування програмного коду в редакторі Bytecode

Робота головного вікна редактора розпочинається зі створювання самого вікна. У програмному коді можна побачити, що ми вказуємо як розмір вікна, так і ім'я. Ім'я – дуже важливий атрибут, бо він не тільки відповідає за «розпізнювання користувачем», а і за системні виклики. Для того щоб ОС могла відрізняти різні, непов'язані між собою програми, вона використовує так звану таблицю імен та ID.

Для оновлення вікна (window repaint), використано стандартну конструкцію роботи ведучої функції main (рис. 2).

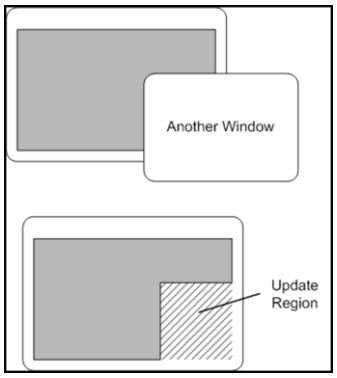

Рис. 2 – Неповне оновлення вікна

Очищення, малювання нового контенту у буффер обміну та відображення нового контенту на екран монітора – це і є стандартний функціонал кожного програмного рішення. Схема, за якою працює текстовий редактор Bytecode така сама (рис. 3).

Можна побачити, що наше програмне вікно не постійно оновлюється без нового контенту або розміру вікна. Завдяки системним сигналам від MS Windows є змога вирахувати зону оновлення вікна, щоб не оновлювати весь інтерфейс, це заощаджує ресурси комп'ютера. Також вікно працює за частотою 60 Hz при допомозі VSync технології. Це було зроблено для того, щоб не витрачати «процесорний час» та не «перегрівати ЦП».

Наведемо принцип роботи редактора Bytecode з інтерфейсом (рис. 4).

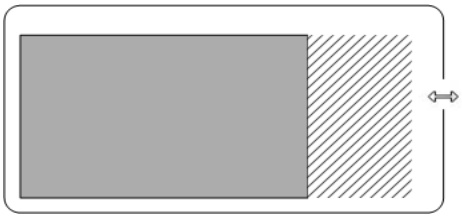

Рис. 3 – Оновлення у разі змінення розміру вікна

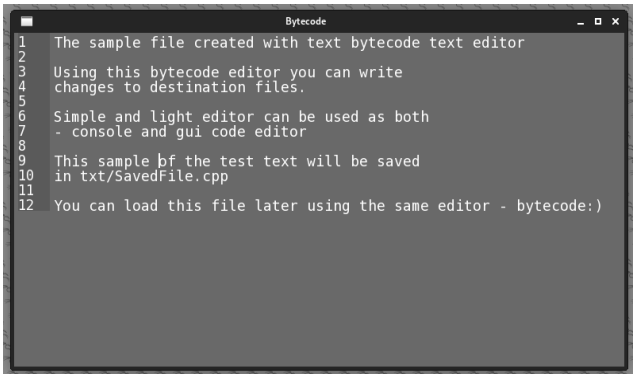

Рис. 4 – Вікно редактора з надрукованим текстом

Процес навігації відбувається за двома послідовностями, а саме за логічною навігацією та програмною. Під логічною навігацією розуміється відстежування індексації кожного пікселя у вікні, курсора, контенту символьних масивів.

Програмна навігація – це те, що було описано раніше, відображення курсора на екрані монітора, оновлення тексту на моніторі та інше. Знаючи на логічному рівні де знаходиться курсор (а ми зберігаємо його дані позиції X та Y), та знаючи висоту і ширину символів у пікселях є змога намалювати курсор у вікні користувача. Такий принцип стосується і всього іншого, що несе в собі редактор Bytecode.

Навігація у редакторі, одне із головних функціоналів, яким повинен володіти редактор програмного коду. Наведемо скріншоти роботи даного функціоналу у редакторі Bytecode (рис. 5–6).

На діаграмі діяльності (рис. 7) показано послідовність актів дій системи на основі діяльності. Діаграми діяльності подібні до процедурних діаграм потоку, але відрізняються від них тим, що діяльності точно прив'язано до об'єктів.

Опис послідовності кожного акту у діаграмі діяльності:

– робота програми розпочинається зі зчитування аргументів із терміналу за допомогою атрибутів головної функції main, а саме argc, argv. Зчитавши ім'я бажаного для відкриття файлу, розпочинаємо роботу текстового редактора;

– для відображення робочого вікна, у якому буде працювати користувач, використовуємо функціонал бібліотеки SFML, а саме починаємо будувати вікно. Запросивши екземпляр вікна з бібліотеки SFML, надаємо наступні параметри: розмір вікна, колір, та назва вікна, як заголовок;

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

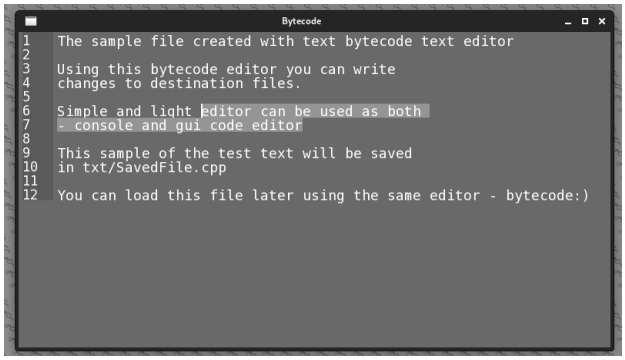

Рис. 5 – Виділення надрукованого тексту у редакторі Bytecode

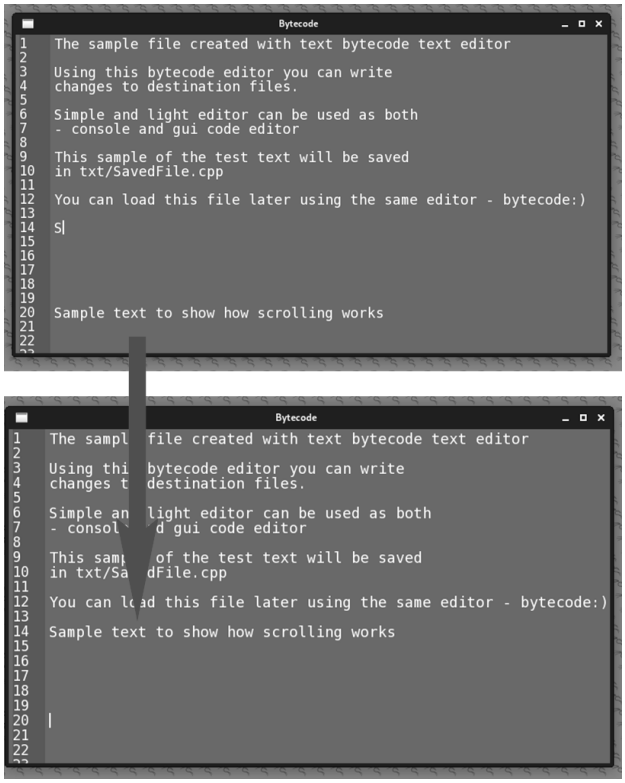

Рис. 6 – Переміщення виділених строк тексту

– створення екземпляра класу TextDocument для подальшої роботи із редагуванням файлу;

– створення екземпляра класу EditorContent – це є обов'язковим, бо цей клас має важливе значення для функціоналу самого редактора. Навігація, відображення, зчитування позиції курсора, виділення тексту – саме за це відповідає цей екземпляр;

– пов'язуємо наш інтерфейс редактора із функціоналом, передаючи у клас EditorView, EditorContent. Клас EditorView відповідає за відображення на екрані загальної програми;

– ініціюємо функціонал класу InputController. Це дуже важливий клас, бо він відповідає не тільки за зчитування тексту з клавіатури, а і за чергу системних повідомлень від редактора. Наприклад, якщо розглядати роботу редактора у MS Window, системні повідомлення є головним, із-за чого працює операційна система MS Windows. Такі системні повідомлення як WM_DESTROY, WM_CLOSE, WM_RESIZE, WM_DRAW та багато інших – лише вони контролюють віконні рішення;

– при натисканні клавіш для збереження змін даних робимо запис файлу або створення його, якщо файл до цього не існував;

– закриття текстового редактора з подальшим звільненням ресурсів робочого вікна редактора.

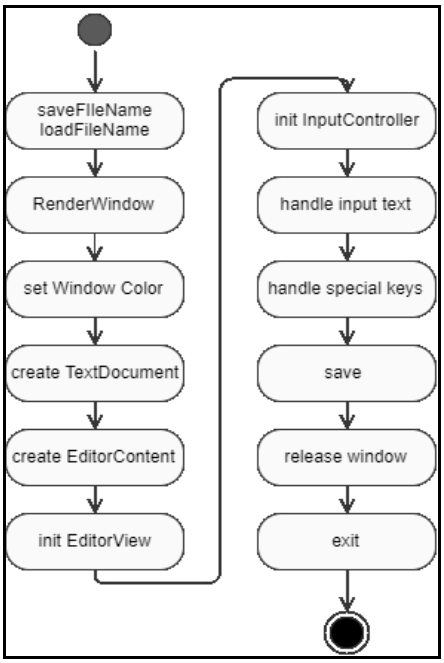

Рис. 7 – Діаграма діяльності

Тестування базувалось на функціональному підході [12–14], дане тестування взаємодіє з іншими системами, і може бути представлено на всіх рівнях тестування. Використання даного типу тестування носить у собі більш практичний підхід до виконання роботи. За таким типом тестування можливо проробити усі сценарії (функціонал, логіку функціоналу) використання з точки зору користувача програмного рішення.

Обговорення результатів

Редактор коду Bytecode – побудований за всіма принципами сучасного об'єктно орієнтованого програмування, що надає змогу проведення як функціонального тестування, так і опису загальної структури за допомогою UML діаграм.

Розробка цього програмного рішення мала більш практичний характер, а саме наведення сутності використання, якомога менше сторонніх бібліотек і використання сучасного системного програмування. Використання великої кількості сторонніх бібліотек є проблемою у сучасному програмуванні, бо за такою структурою важко описати дії програми та побудувати прозорий алгоритм дії.

Провівши тестування редактора на функціонал та сумісність із різними ОС, можна помітити, що основним вимогам редактор повністю відповідає (табл. 1).

Таблиця 1 – Перевірка роботи редактора на сумісність в ОС

Операційна система	GCC 4.9.2 Windows	GCC 4.x.x Linux	Visual C++ 12
Linux Ubuntu	-	+	-
Linux Mint	-	+	-
Kali Linux	-	+	-
MS Windows 10	+	-	+
MS Windows 8	+	-	+
MS Windows 7	+	-	+

Такий функціонал як відображення на екран та редагування програмного тексту пройшов тестування на обох операційних системах: UNIX та MS Windows.

Збереження у файл – працюючий функціонал, який був побудований на базі можливостей мови C/C++. Маючи такі функції як побітове зчитування з документу, мова C/C++ вирішує проблему використання сторонніх технологій та включення бібліотек, які більш виконують роль абстракції і не завжди несуть оптимізований програмний код.

Задля використання редактора на відповідній ОС, потрібно мати сумісні технології для компілювання текстового редактора Bytecode. За табл. 1 можна побачити, що використання одного і того ж компілятора не є вирішенням проблеми роботи редактора на обох операційних системах. Використання задокументованого компілятора, допоможе уникнути втрати функціоналу або взагалі повної роботоздатності програми. Сумісні версії компіляторів можна, наприклад, перевірити на сайті бібліотеки SFML.

Висновки

У даній роботі розглядалась задача розробки текстового редактора для змінення та збереження інформації.

За результатами виконання роботи:

– реалізовано базовий функціонал редактора (збереження, змінення);

– реалізовано сумісність редактора з різними операційними системами;

– розроблено як графічний, так і консольний інтерфейс програми;

– розроблена спеціальна архітектура програмного забезпечення, яка дозволяє підтримувати та удосконалювати програмне забезпечення у майбутньому;

– реалізовано простий і зрозумілий інтерфейс;

– результати роботи програмного забезпечення були протестовані для різних вхідних даних.

Були використані наступні мови програмування: C, C++11.

Визначено та реалізовано метод обробки вхідної інформації, а також збереження у файл окремого формату. Розроблена зручна архітектура програмного забезпечення, яка дозволяє з легкістю підтримувати та удосконалювати програму у майбутньому. Реалізовано дружній графічний інтерфейс для взаємодії з користувачем.

Список літератури

1. Duffy Michael D. Chapter 13 - Text Editors. Getting Started with OpenVMS, edited by Michael D. Duffy. *Digital Press.* 2003. P. 167–175. doi:10.1016/B978-155558279-1/50014-2.
2. Kidwai Abdullah, et al. A Comparative Study on Shells in Linux: A Review. *Materials Today: Proceedings.* Jan. 2021. Vol. 37. P. 2612–16. doi: 10.1016/j.matpr.2020.08.508.
3. Smith David W. Chapter 2 - First C Program. PIC Projects and Applications Using C (Third Edition), edited by David W. Smith. *Newnes.* 2013. P. 5–34. doi: 10.1016/B978-0-08-097151-3.00002-7.
4. Dai Peng, et al. An Improving Approach to Analyzing Change Impact of C Programs. *Computer Communications*. Jan. 2022. Vol. 182. P. 60–71. doi: 10.1016/j.comcom.2021.10.039.
5. Kang Kyuchang, et al. SFML: Screening Form Markup Language for Healthcare Service. *2012 14th International Conference on Advanced Communication Technology (ICACT).* 2012. P. 1284–88.
6. Clercq Anton le and Kristoffer Almroth. *Comparison of Rendering Performance Between Multimedia Libraries Allegro, SDL and SFML*. 2019. 52 p.
7. Khawaja Gus. Linux Privilege Escalation. Kali Linux Penetration Testing Bible. *Wiley*. 2021. P. 257–272. URL: https://www.wiley.com/en-us/Kali+Linux+Penetration+Testing+Bible-p-9781119719083.
8. Xiao Guanping, et al. Evolution of Linux Operating System Network. *Physica A: Statistical Mechanics and Its Applications*. Jan. 2017. Vol. 466. P. 249–58. doi: 10.1016/j.physa.2016.09.021.
9. Bourne Kelly C. Chapter 25 - Linux Tools. *Application Administrators Handbook, edited by Kelly C. Bourne, Morgan Kaufmann.* 2014. P. 457–63. doi: 10.1016/B978-0-12-398545-3.00025-X.
10. Beuchelt Gerald. Chapter 11 - UNIX and Linux Security. *Computer and Information Security Handbook (Third Edition), edited by John R. Vacca, Morgan Kaufmann*. 2017. P. 205–24. doi: 10.1016/B978-0-12-803843-7.00011-9.
11. Abbott Doug. Chapter 3 - Introducing Linux. *Linux for Embedded and Real-Time Applications (Fourth Edition), edited by Doug Abbott, Newnes*. 2018. P. 29–54. doi: 10.1016/B978-0-12-811277-9.00003-1.
12. Baresi Luciano and Mauro Pezzè. An Introduction to Software Testing. *Electronic Notes in Theoretical Computer Science*. Feb. 2006. Vol. 148, no. 1. P. 89–111. doi: 10.1016/j.entcs.2005.12.014.
13. Howden William E. Functional Testing and Design Abstractions. *Journal of Systems and Software*. Jan. 1979. Vol. 1. P. 307–13. doi: 10.1016/0164-1212(79)90032-3.
14. Garousi Vahid, et al. Software-Testing Education: A Systematic Literature Mapping. *Journal of Systems and Software*. July 2020. Vol. 165. P. 110570. doi: 10.1016/j.jss.2020.110570.

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

References (transliterated)

1. Duffy Michael D. Chapter 13 - Text Editors. Getting Started with OpenVMS, edited by Michael D. Duffy. *Digital Press,* 2003, pp. 167–175, doi:10.1016/B978-155558279-1/50014-2.
2. Kidwai Abdullah, et al. A Comparative Study on Shells in Linux: A Review. *Materials Today: Proceedings,* Jan. 2021, vol. 37, pp. 2612–16, doi: 10.1016/j.matpr.2020.08.508.
3. Smith David W. Chapter 2 - First C Program. PIC Projects and Applications Using C (Third Edition), edited by David W. Smith. *Newnes,* 2013, pp. 5–34, doi: 10.1016/B978-0-08-097151-3.00002-7.
4. Dai Peng, et al. An Improving Approach to Analyzing Change Impact of C Programs. *Computer Communications,* Jan. 2022, vol. 182, pp. 60–71, doi: 10.1016/j.comcom.2021.10.039.
5. Kang Kyuchang, et al. SFML: Screening Form Markup Language for Healthcare Service. *2012 14th International Conference on Advanced Communication Technology (ICACT)*. 2012, pp. 1284–88.
6. Clercq Anton le and Kristoffer Almroth. *Comparison of Rendering Performance Between Multimedia Libraries Allegro, SDL and SFML*. 2019, 52 p.
7. Khawaja Gus. Linux Privilege Escalation. Kali Linux Penetration Testing Bible. *Wiley*, 2021, pp. 257–272. Available at: https://www.wiley.com/en-us/Kali+Linux+Penetration+Testing+Bible-p-9781119719083.
8. Xiao Guanping, et al. Evolution of Linux Operating System Network. *Physica A: Statistical Mechanics and Its Applications,* Jan. 2017, vol. 466, pp. 249–58, doi: 10.1016/j.physa.2016.09.021.
9. Bourne Kelly C. Chapter 25 - Linux Tools. *Application Administrators Handbook, edited by Kelly C. Bourne, Morgan Kaufmann*, 2014, pp. 457–63, doi: 10.1016/B978-0-12-398545-3.00025-X.
10. Beuchelt Gerald. Chapter 11 - UNIX and Linux Security. *Computer and Information Security Handbook (Third Edition), edited by John R. Vacca, Morgan Kaufmann,* 2017, pp. 205–24, doi: 10.1016/B978-0-12-803843-7.00011-9.
11. Abbott Doug. Chapter 3 - Introducing Linux. *Linux for Embedded and Real-Time Applications (Fourth Edition), edited by Doug Abbott, Newnes*, 2018, pp. 29–54, doi: 10.1016/B978-0-12-811277-9.00003-1.
12. Baresi Luciano, and Mauro Pezzè. An Introduction to Software Testing. *Electronic Notes in Theoretical Computer Science,* Feb. 2006, vol. 148, no. 1, pp. 89–111, doi: 10.1016/j.entcs.2005.12.014.
13. Howden William E. Functional Testing and Design Abstractions. *Journal of Systems and Software*, Jan. 1979, vol. 1, pp. 307–13, doi: 10.1016/0164-1212(79)90032-3.
14. Garousi Vahid, et al. Software-Testing Education: A Systematic Literature Mapping. *Journal of Systems and Software*, July 2020, vol. 165, p. 110570, doi: 10.1016/j.jss.2020.110570.

Відомості про авторів (About authors)

Мєтєльов Володимир Олександрович – кандидат технічних наук, Національний технічний університет «Харківський політехнічний інститут», доцент кафедри комп'ютерного моделювання процесів та систем; м. Харків, Україна; ORCID: https://orcid.org/0000-0002-2633-6296; e-mail: volodymyr.mietielov@khpi.edu.ua.

Mietielov Volodymyr – Ph. D., Associate Professor of the Department of Computer Modelling of Processes and Systems, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0002-2633-6296; e-mail: volodymyr.mietielov@khpi.edu.ua.

Марусенко Олексій Миколайович – Національний технічний університет «Харківський політехнічний інститут», асистент кафедри комп'ютерного моделювання процесів та систем; м. Харків, Україна; ORCID: https://orcid.org/0000-0001-6911-2500; e-mail: Oleksii.Marusenko@khpi.edu.ua.

Marusenko Oleksii – Assistant of the Department of Computer Modelling of Processes and Systems, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0001-6911-2500; e-mail: Oleksii.Marusenko@khpi.edu.ua.

Баскаков Олександр Романович – Національний технічний університет «Харківський політехнічний інститут», студент кафедри комп'ютерного моделювання процесів та систем; м. Харків, Україна; e-mail: baskakovforjob@gmail.com.

Baskakov Oleksandr – Student of the Department of Computer Modelling of Processes and Systems, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine; e-mail: baskakovforjob@gmail.com.

Будь ласка, посилайтесь на цю статтю наступним чином:

Мєтєльов В. О., Марусенко О. М., Баскаков О. Р. Розробка текстового редактора для операційної системи UNIX. *Вісник Національного технічного університету «ХПІ». Серія: Нові рішення в сучасних технологіях*. – Харків: НТУ «ХПІ». 2022. № 4 (14). С. 35-40. doi:10.20998/2413-4295.2022.04.05.

Please cite this article as:

Mietielov V., Marusenko O., Baskakov O. Development of a text editor for UNIX operating system. *Bulletin of the National Technical University "KhPI". Series: New solutions in modern technology*. – Kharkiv: NTU "KhPI", 2022, no. 4(14), pp. 35–40, doi:10.20998/2413-4295.2022.04.05.

Надійшла (received) 02.12.2022