МІНІСТЕРСТВО ОСВІТИ І НАУКИ
ДЕРЖАВНИЙ ЗАКЛАД
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Г.А. Могильний, О.О. Смагіна, С. О. Переяславська

ОПЕРАЦІЙНІ СИСТЕМИ ТА СИСТЕМНЕ ПРОГРАМУВАННЯ (ЧАСТИНА 1)

Методичні рекомендації до виконання лабораторних робіт з освітнього компонента для здобувачів освіти спеціальності 122– „Комп’ютерні науки”

Полтава
ДЗ „ЛНУ імені Тараса Шевченка”
2024

УДК 004.45+004.41(072)
М74

Р е ц е н з е н т и :

Козуб Ю.Г. – доктор технічних наук, професор, в.о. завідувача кафедри математики та інформатики ДЗ „Луганський національний університет імені Тараса Шевченка”.

Лахно В.А. – доктор технічних наук, професор кафедри комп'ютерних систем, мереж та кібербезпеки, Національний університет біоресурсів і природокористування України, м. Київ.

М74

Могильний Г.А. Операційні системи та системне програмування (частина 1) : Методичні рекомендації до виконання лабораторних робіт з освітнього компонента для здобувачів освіти спеціальності 122 – Комп’ютерні науки / Г.А. Могильний, О. О. Смагіна, С. О. Переяславська; Держ. закл. „Луган. нац. ун-т імені Тараса Шевченка”. – Полтава : ДЗ „ЛНУ імені Тараса Шевченка”, 2024. – 85 с.

Методичні рекомендації структуровано відповідно до розділів робочої програми курсу „Операційні системи та системне програмування” для спеціальності 122 – Комп’ютерні науки кафедри математики та інформатики ДЗ ЛНУ імені Тараса Шевченка. Методичні рекомендації охоплюють вивчення структур та функцій системного програмного забезпечення, призначення, функцій і загальних структурних рішень побудови операційних систем, дослідження операційних систем та систем програмування, одержання навичок роботи у різних операційних системах та системах програмування.

Курс лекцій призначений для здобувачів освіти спеціальності 122 – Комп’ютерні науки.

УДК 004.45+004.41(072)

Рекомендовано до друку Вченою радою
Луганського національного університету імені Тараса Шевченка
(протокол № 10 від 20 грудня 2024 р.)

ЗМІСТ

Модуль 1.

Лабораторна робота 1. Встановлення Linux на віртуальну машину.

Мета роботи: навчитися встановлювати за допомогою віртуальної машини операційні системи

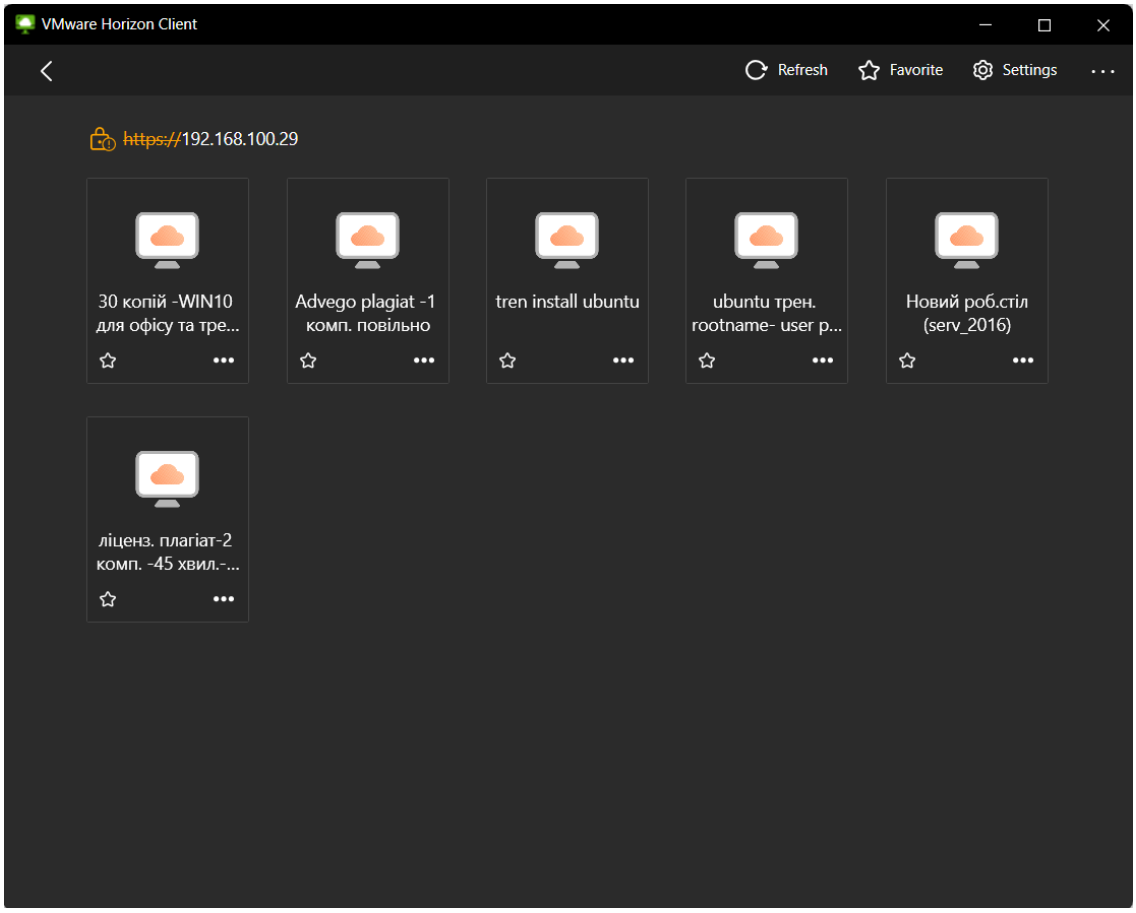
Хід роботи:

1. Ознайомтеся з додатковим матеріалом з лабораторної роботи:

2. Усі етапи роботи описати у звіті з відповідними скріншотами.

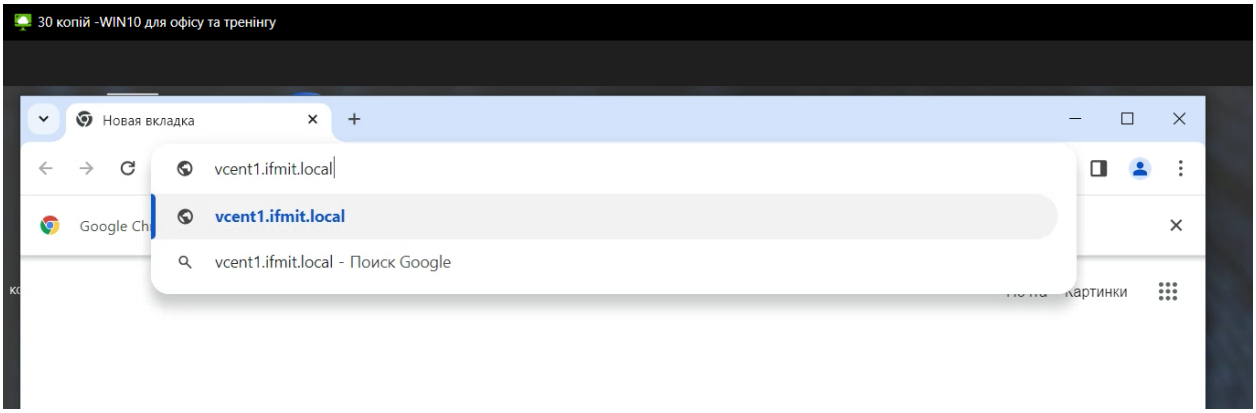
3. Відповісти на контрольні питання.

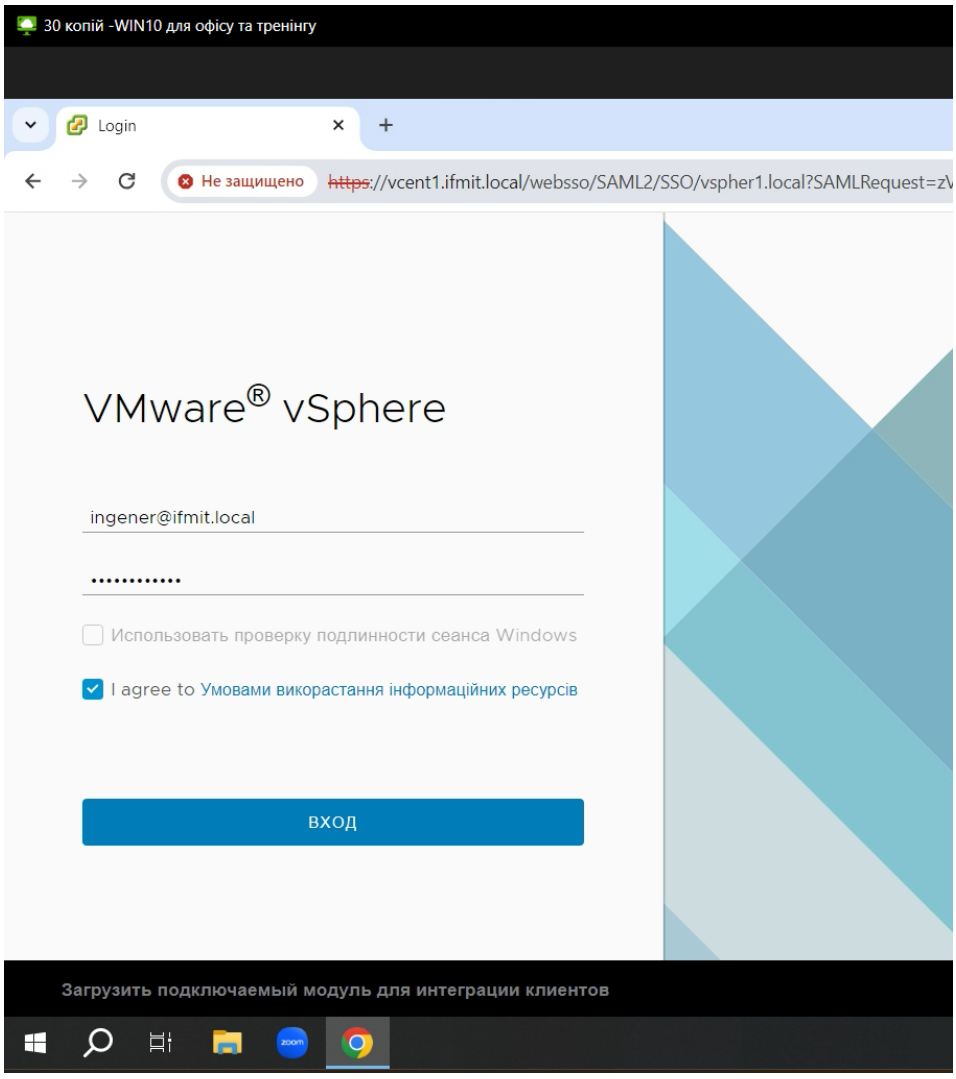
Встановити та зайти в VMware Horizon Client



Приєднуватися до серверу потрібно за адресою **176.105.199.98.4432** з вашим логіном та зайти на першу іконку «30 копій WIN10»

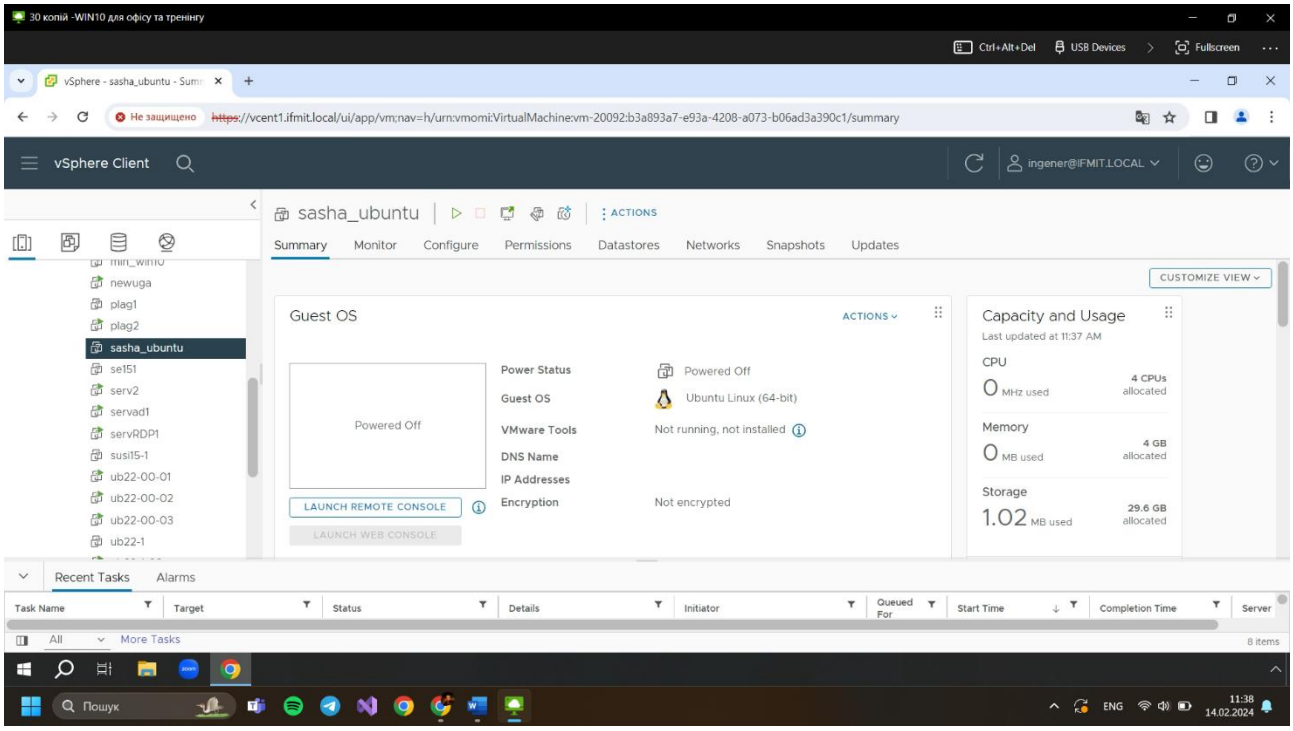
Ви знаходитесь на віртуальній машині. Далі заходите в Chrome та набираєте у пошуковий рядок «**vcent1.ifmit.local**»



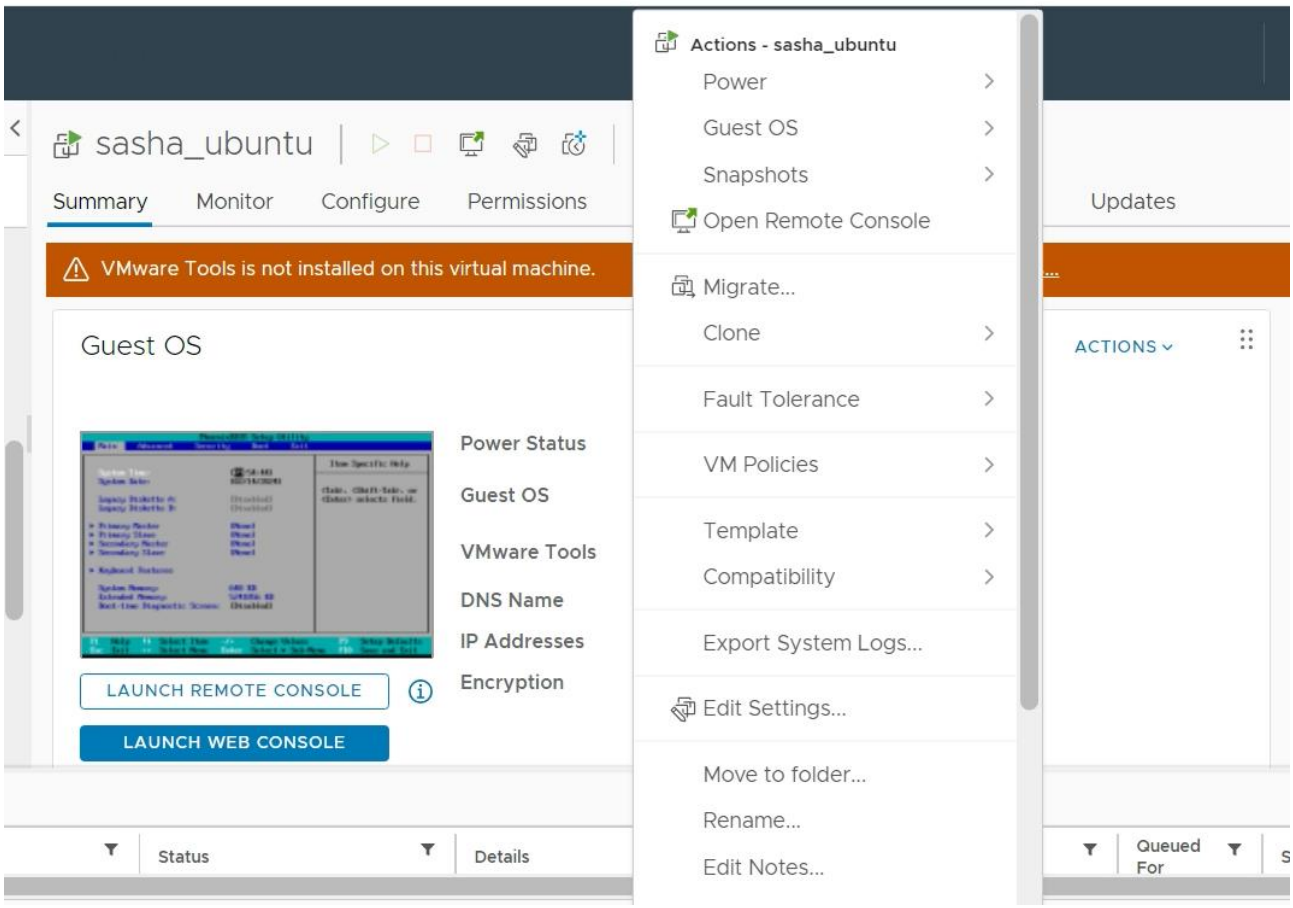


Логінимося на VMware vShape

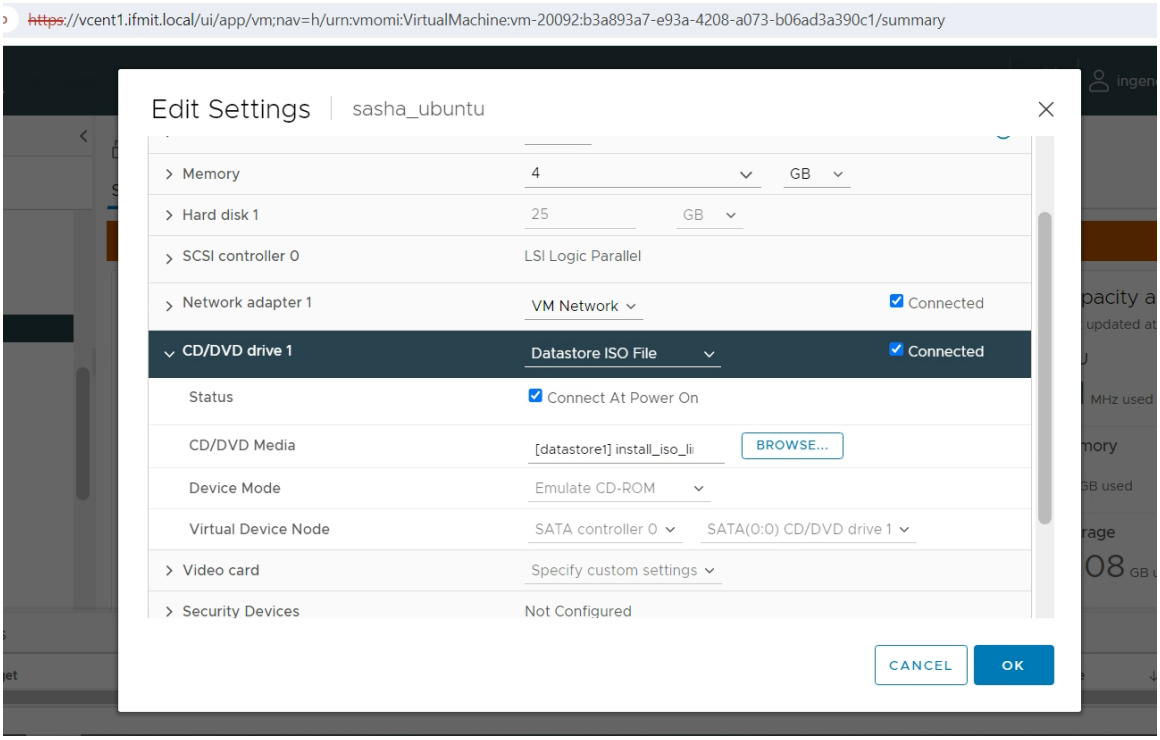
Ви маєте ввести ваш актуальний логін та пароль.



Знаходимо віртуальну машину створену адміном з назвою «**sasha_ubuntu**», заходимо на неї

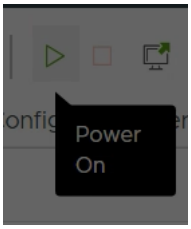


Далі натискаємо «Actions» та заходимо в «Edit Settings…»



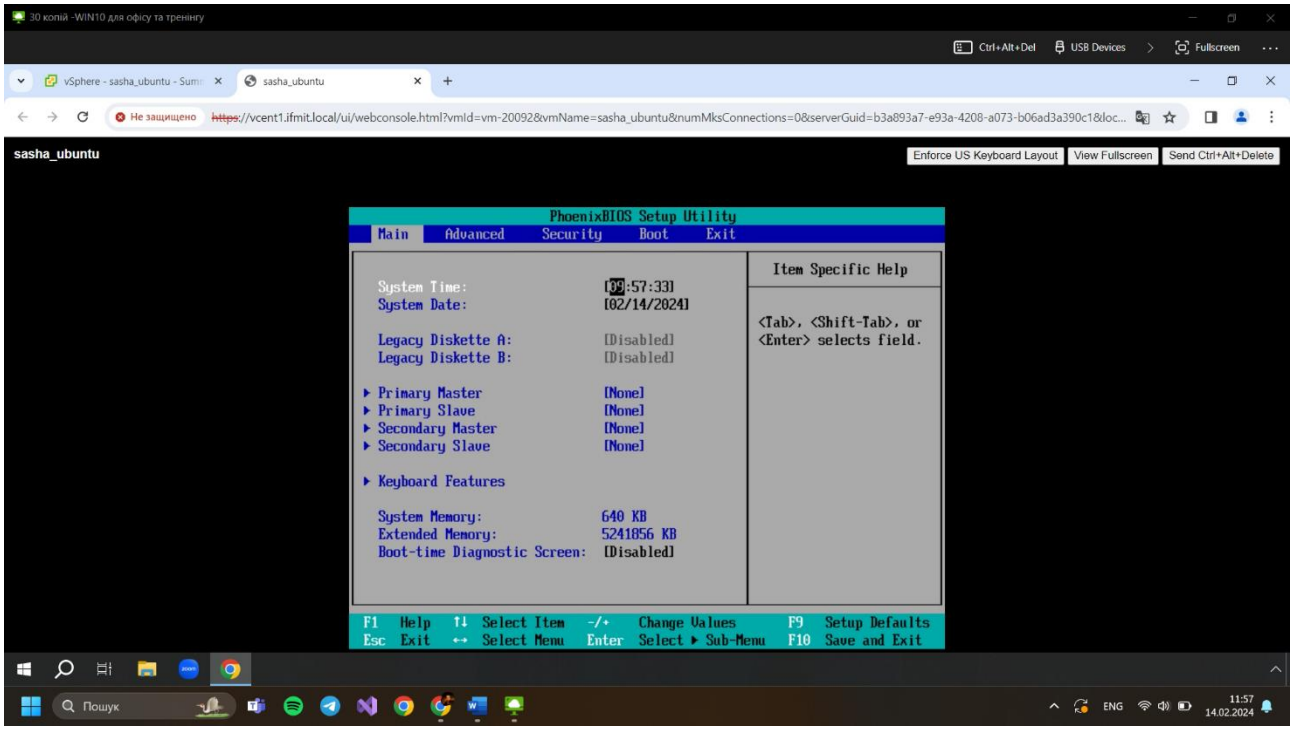
Далі натискаємо на галочку «**connected**» в пункті «**CD/DVD drive 1**»

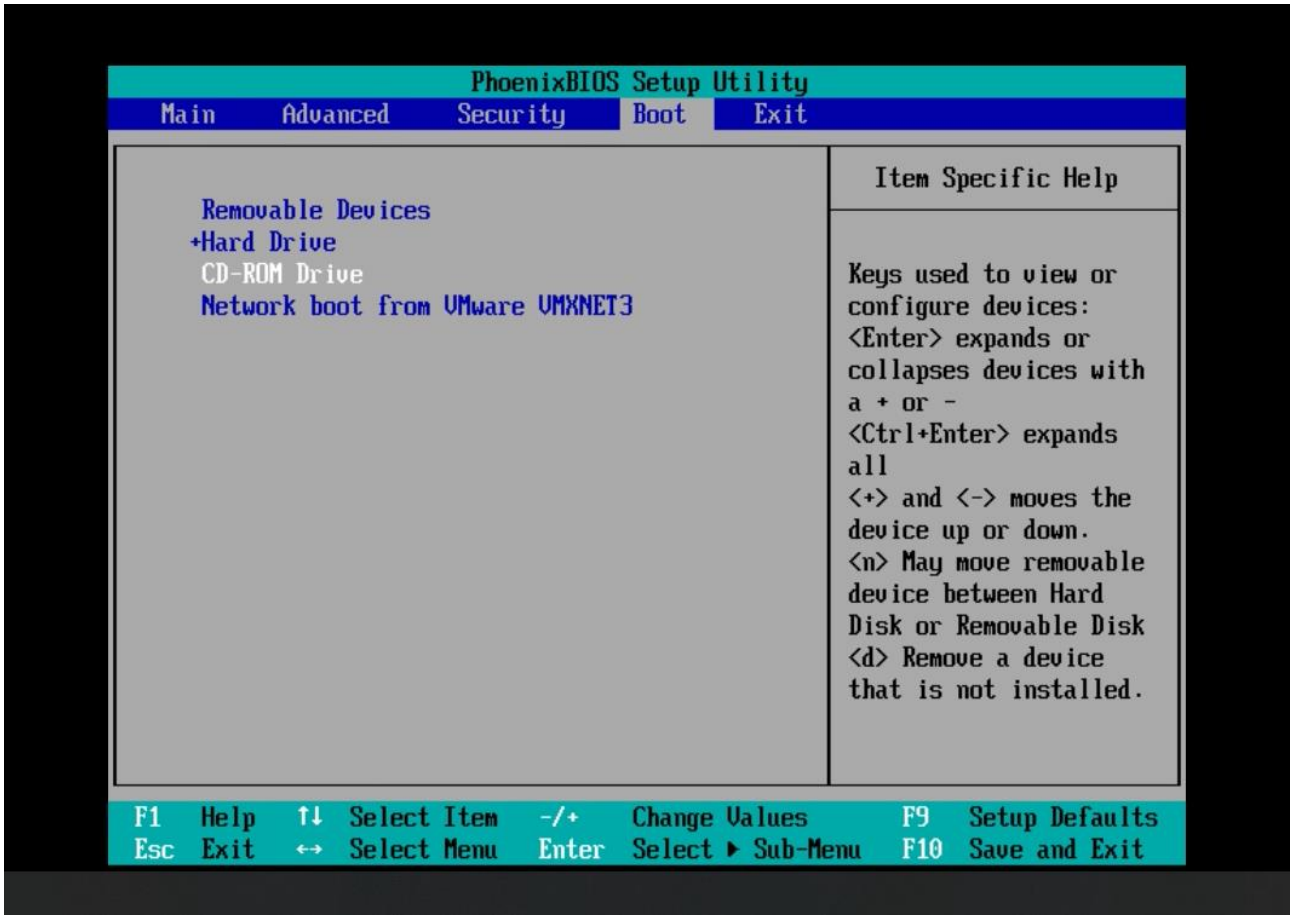
Отже, таким чином підключено віртуальний накопичувач з ISO файлом (образом ubuntu)



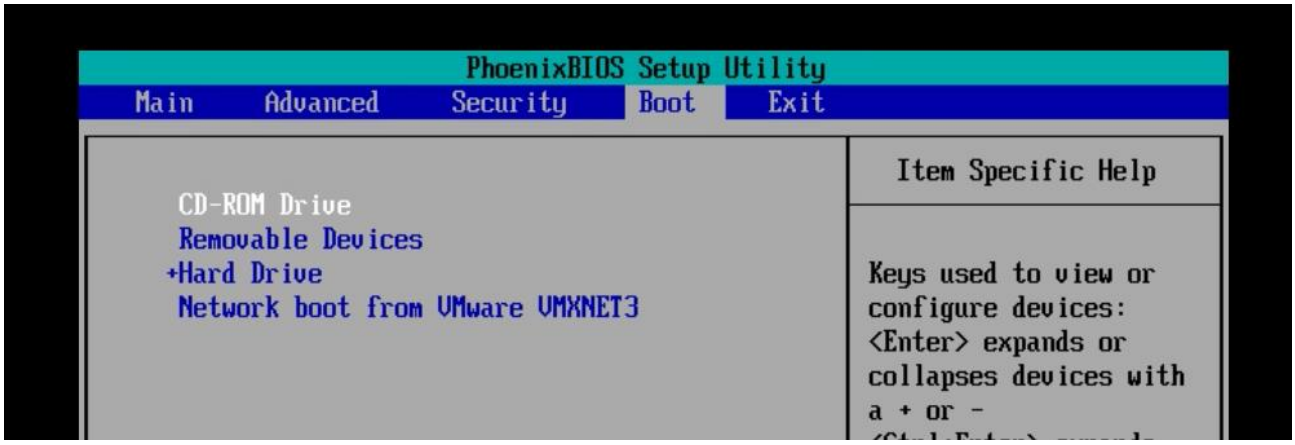
Після підключення віртуального диску, запускаєте машину

І натискаєте «Launch web console» тим самим ми відкриваєте нову вкладку в браузері, де буде відображена ваша віртуальна машина, тільки вже Ubuntu. Тим самим ми маєте віртуальну машину в віртуальній машині.

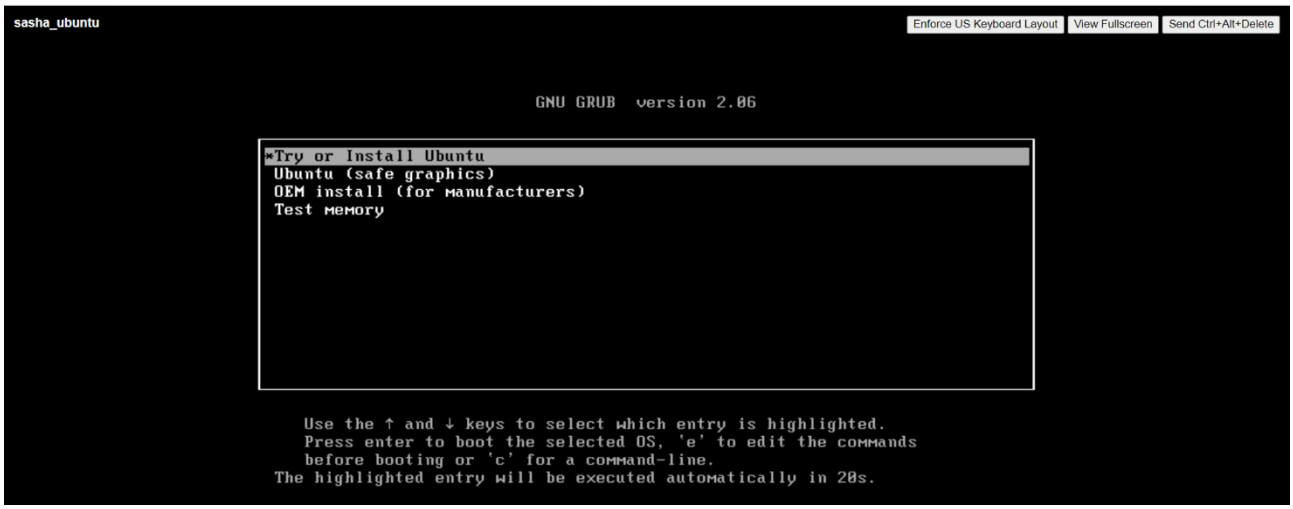




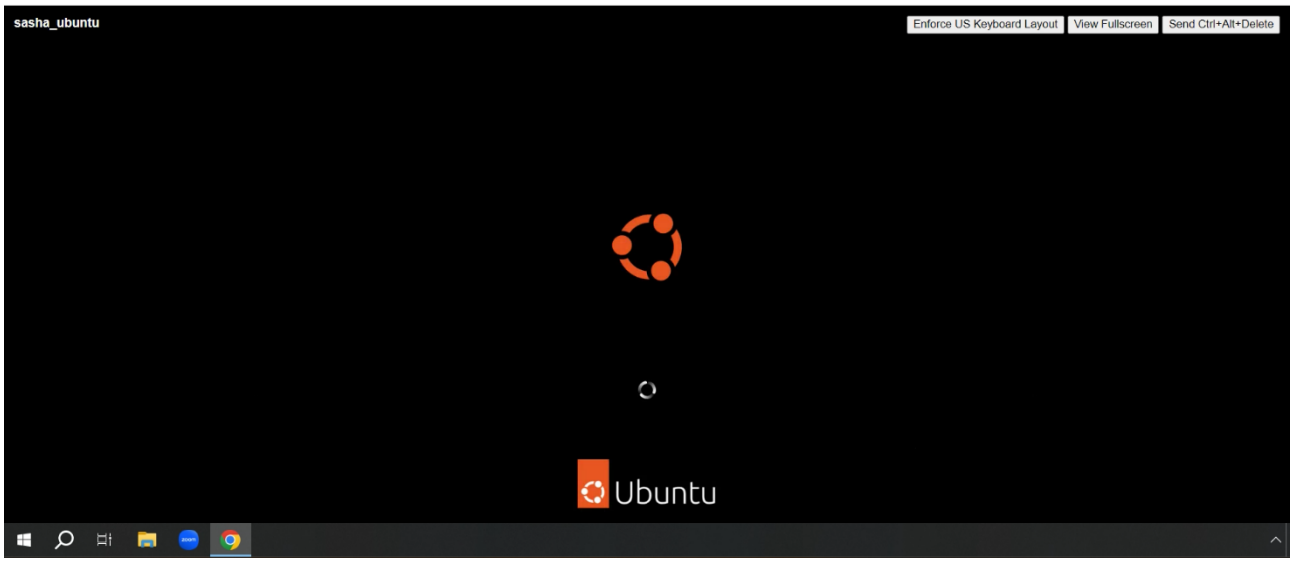
Переходите на вкладку, що на рисунку обираєте наш CD-ROM Drive та перетягуєте натисканням «Shift та +» на перше місце в списку

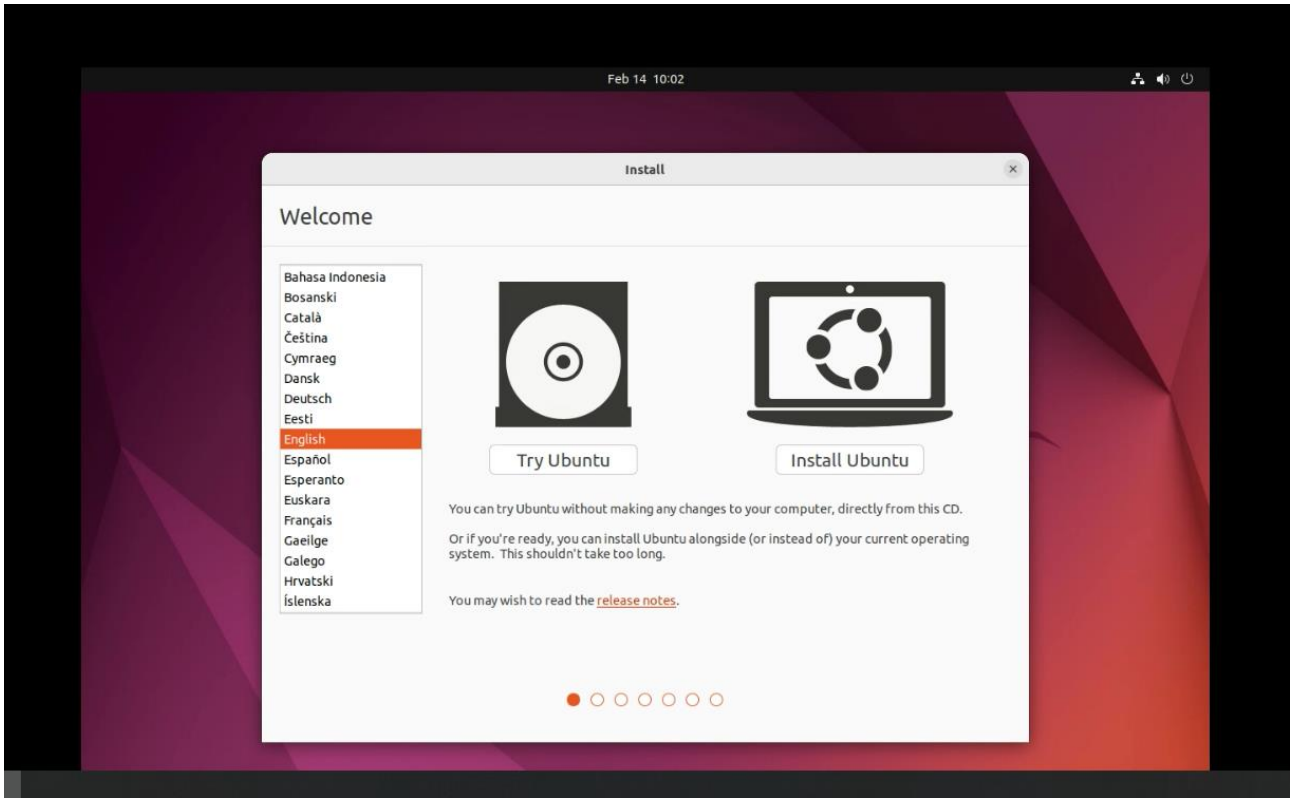


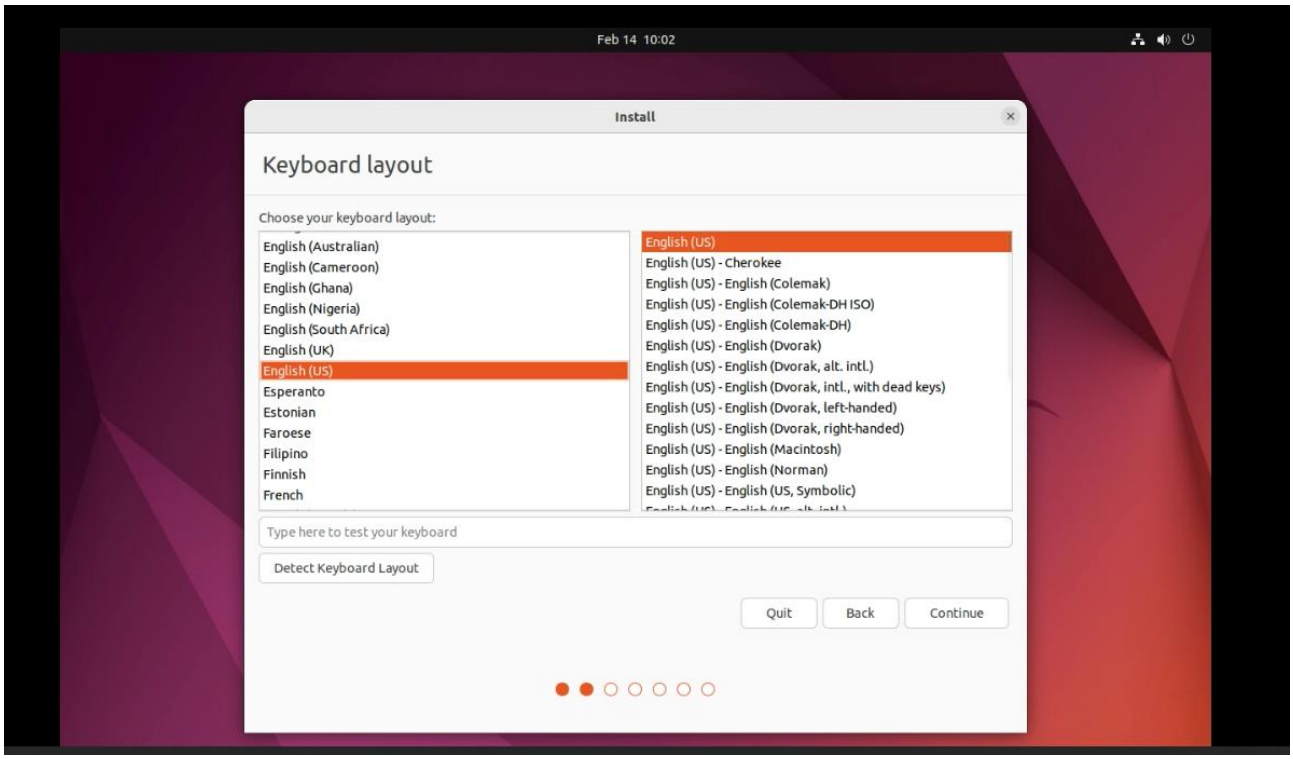
Щоб був такий вигляд. Це потрібно для того, щоб почати завантаження ISO файлу (це потрібно для першого запуску) і перезавантажуєте машину.

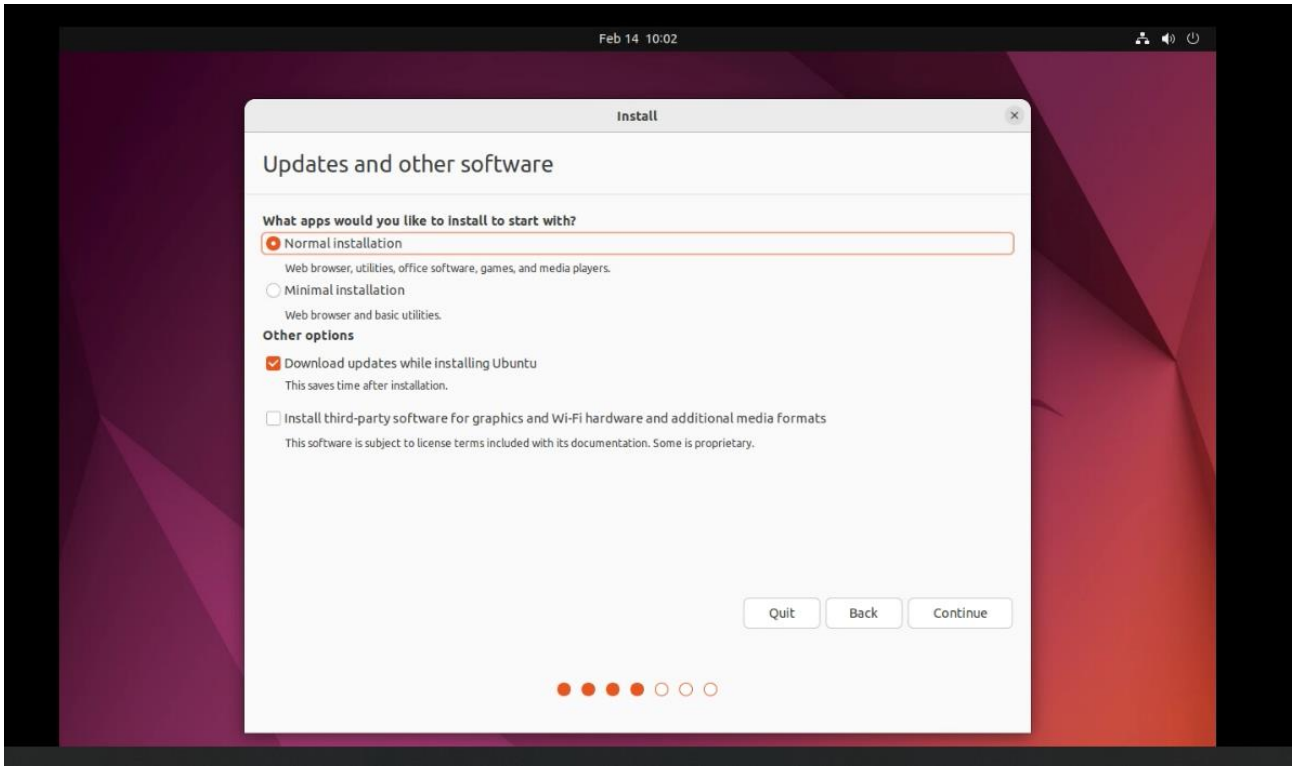


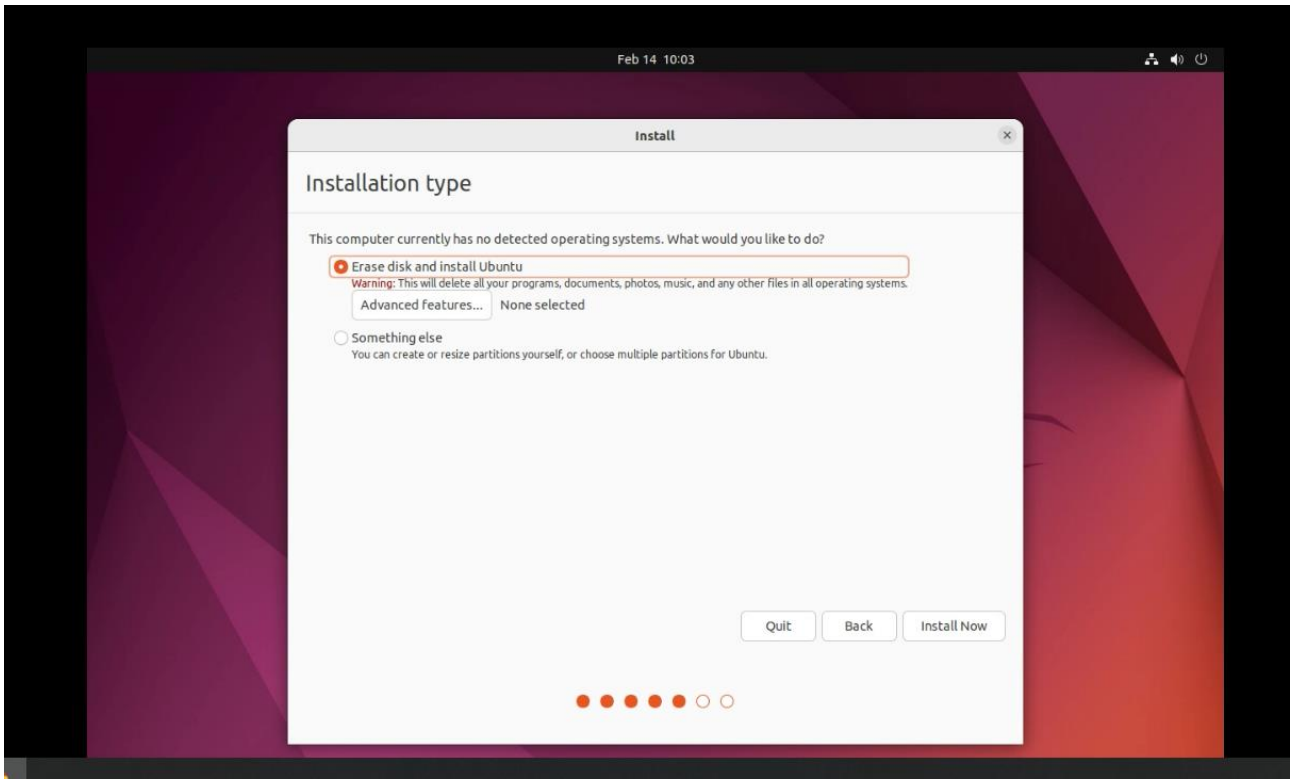
Далі слідуєте за скріншотами

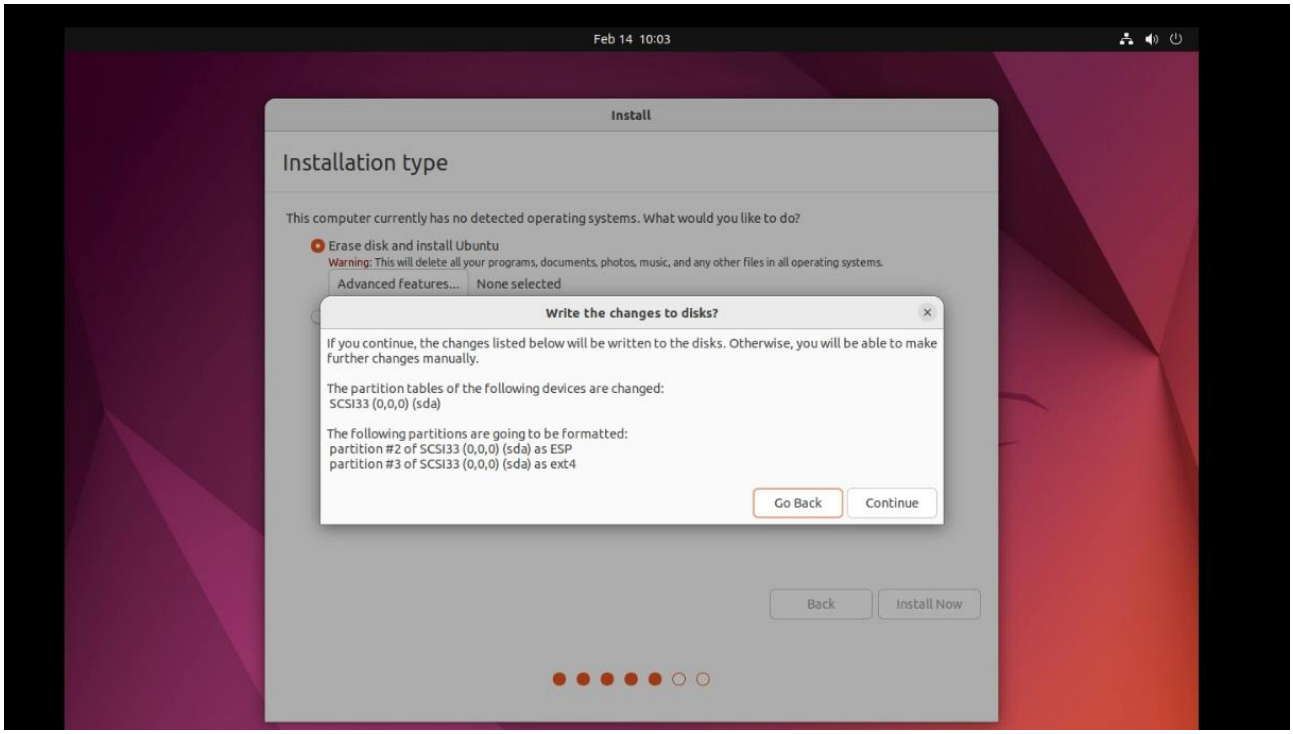


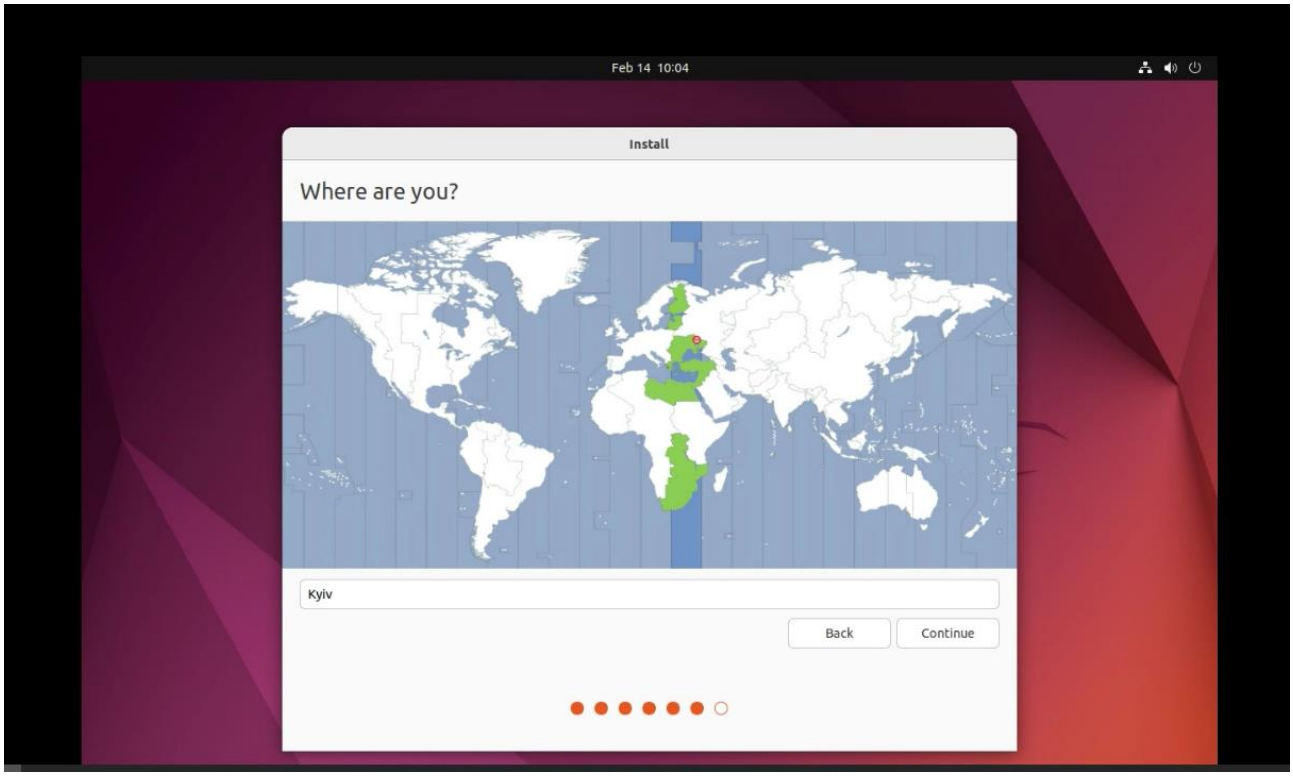

Feb 14 10:02
Install
Welcome
Bahasa Indonesia
Bosanski
Català
Čeština
Cymraeg
Dansk
Deutsch
Eesti
English
Español
Esperanto
Euskara
Français
Gaeilge
Galego
Hrvatski
Íslenska
Try Ubuntu
Install Ubuntu
You can try Ubuntu without making any changes to your computer, directly from this CD.
Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.
You may wish to read the release notes.

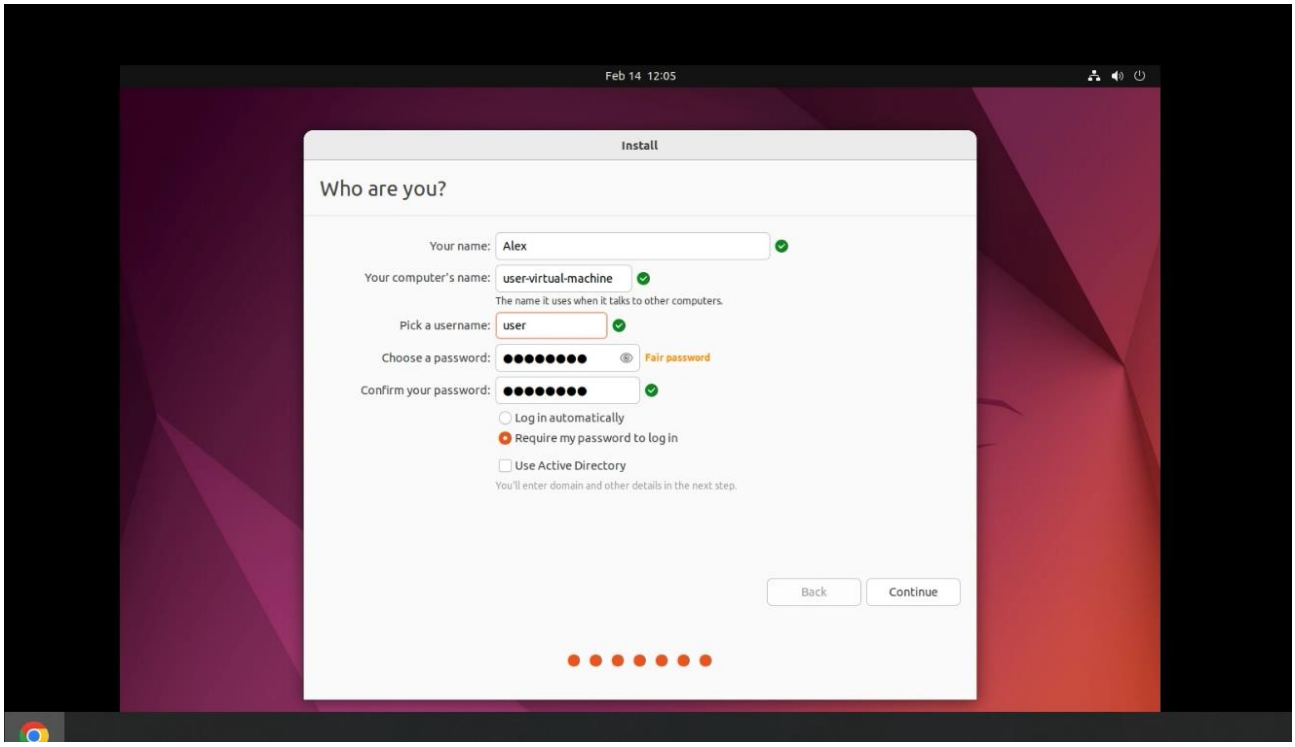

Feb 14 10:02
Install
Keyboard layout
Choose your keyboard layout:
English (Australian)
English (Cameroon)
English (Ghana)
English (Nigeria)
English (South Africa)
English (UK)
English (US)
Esperanto
Estonian
Faroese
Filipino
Finnish
French
English (US)
English (US) - Cherokee
English (US) - English (Colemak)
English (US) - English (Colemak-DH ISO)
English (US) - English (Colemak-DH)
English (US) - English (Dvorak)
English (US) - English (Dvorak, alt. intl.)
English (US) - English (Dvorak, intl., with dead keys)
English (US) - English (Dvorak, left-handed)
English (US) - English (Dvorak, right-handed)
English (US) - English (Macintosh)
English (US) - English (Norman)
English (US) - English (US, Symbolic)
Type here to test your keyboard
Detect Keyboard Layout
Quit
Back
Continue


Feb 14 10:02
Install
Updates and other software
What apps would you like to install to start with?
Normal installation
Web browser, utilities, office software, games, and media players.
Minimal installation
Web browser and basic utilities.
Other options
Download updates while installing Ubuntu
This saves time after installation.
Install third-party software for graphics and Wi-Fi hardware and additional media formats
This software is subject to license terms included with its documentation. Some is proprietary.
Quit
Back
Continue

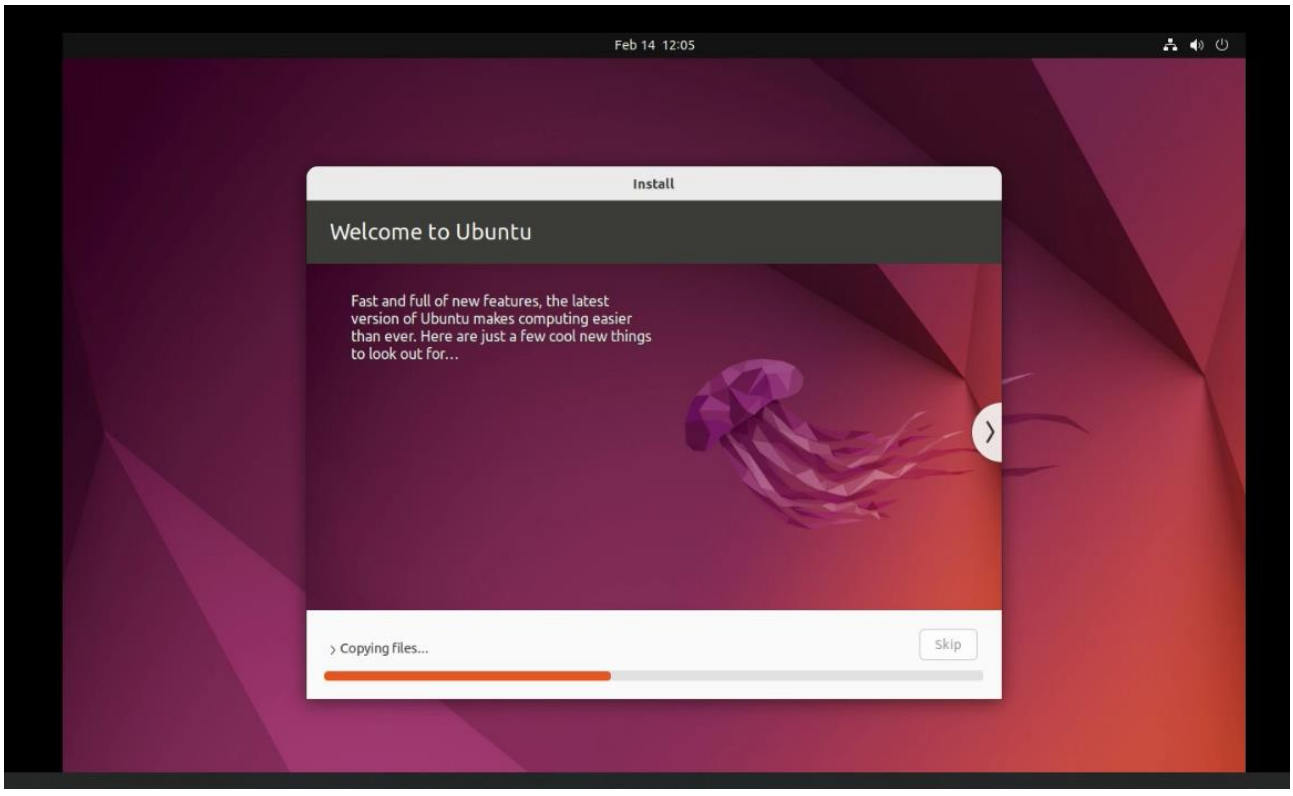

Feb 14 10:03
Install
Installation type
This computer currently has no detected operating systems. What would you like to do?
Erase disk and install Ubuntu
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.
Advanced features...
None selected
Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.
Quit
Back
Install Now

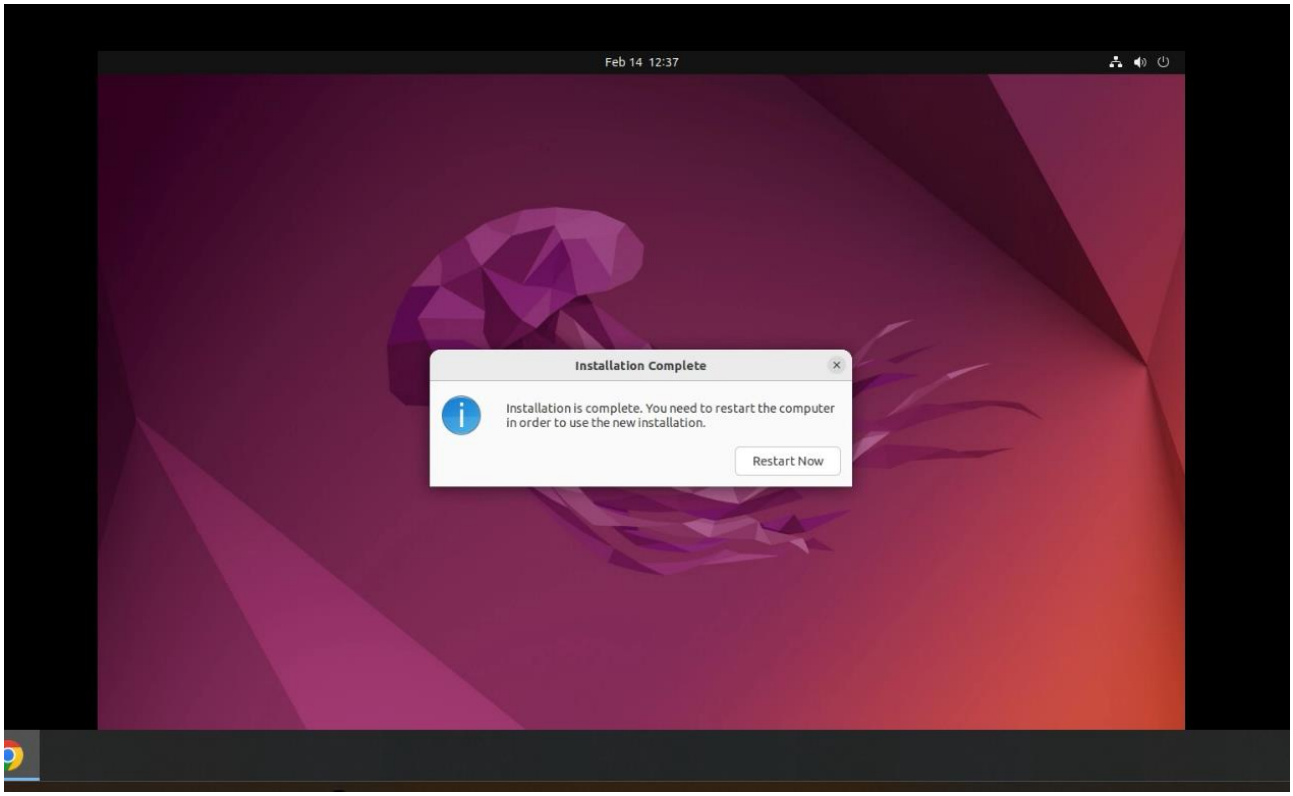

Feb 14 10:03
Install
Installation type
This computer currently has no detected operating systems. What would you like to do?
Erase disk and install Ubuntu
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.
Advanced features...
None selected
Write the changes to disks?
If you continue, the changes listed below will be written to the disks. Otherwise, you will be able to make further changes manually.
The partition tables of the following devices are changed:
SCSI33 (0,0,0) (sda)
The following partitions are going to be formatted:
partition #2 of SCSI33 (0,0,0) (sda) as ESP
partition #3 of SCSI33 (0,0,0) (sda) as ext4
Go Back
Continue
Back
Install Now

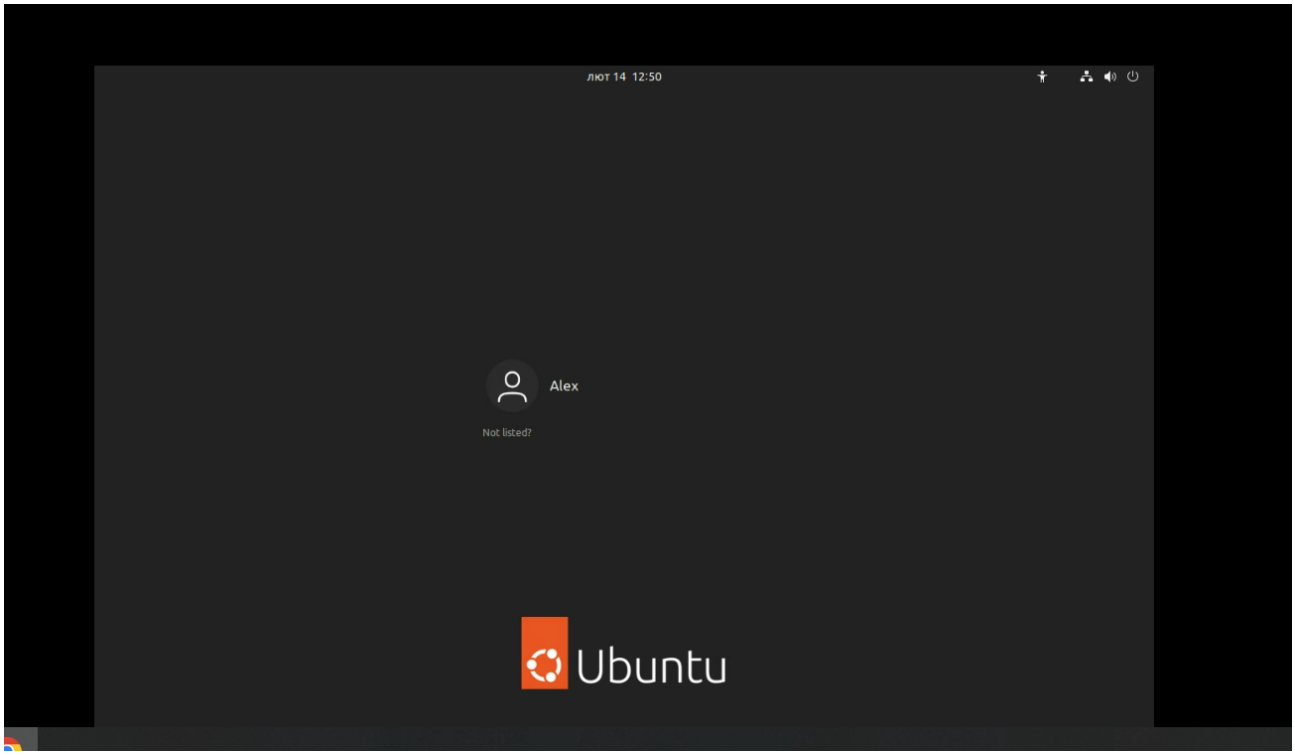

Feb 14 10:04
Install
Where are you?
Kyiv
Back
Continue



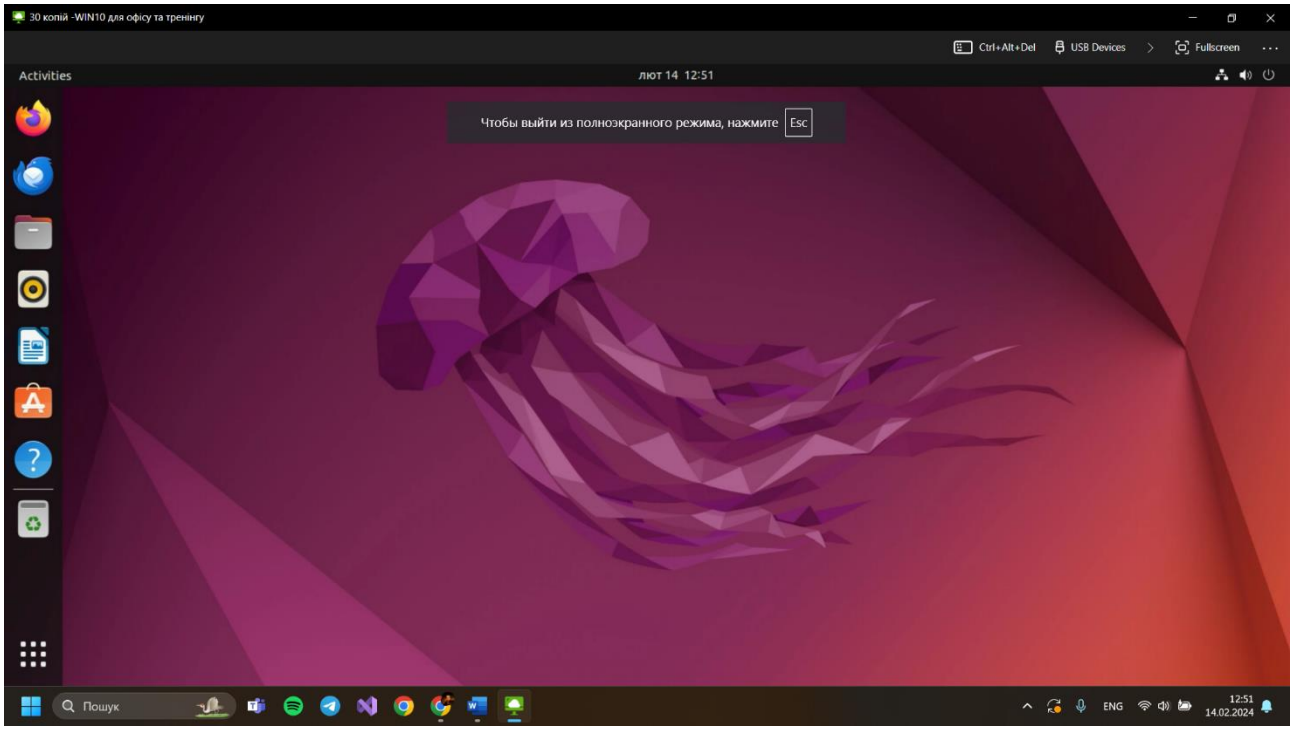
Створюєте адмін акаунт

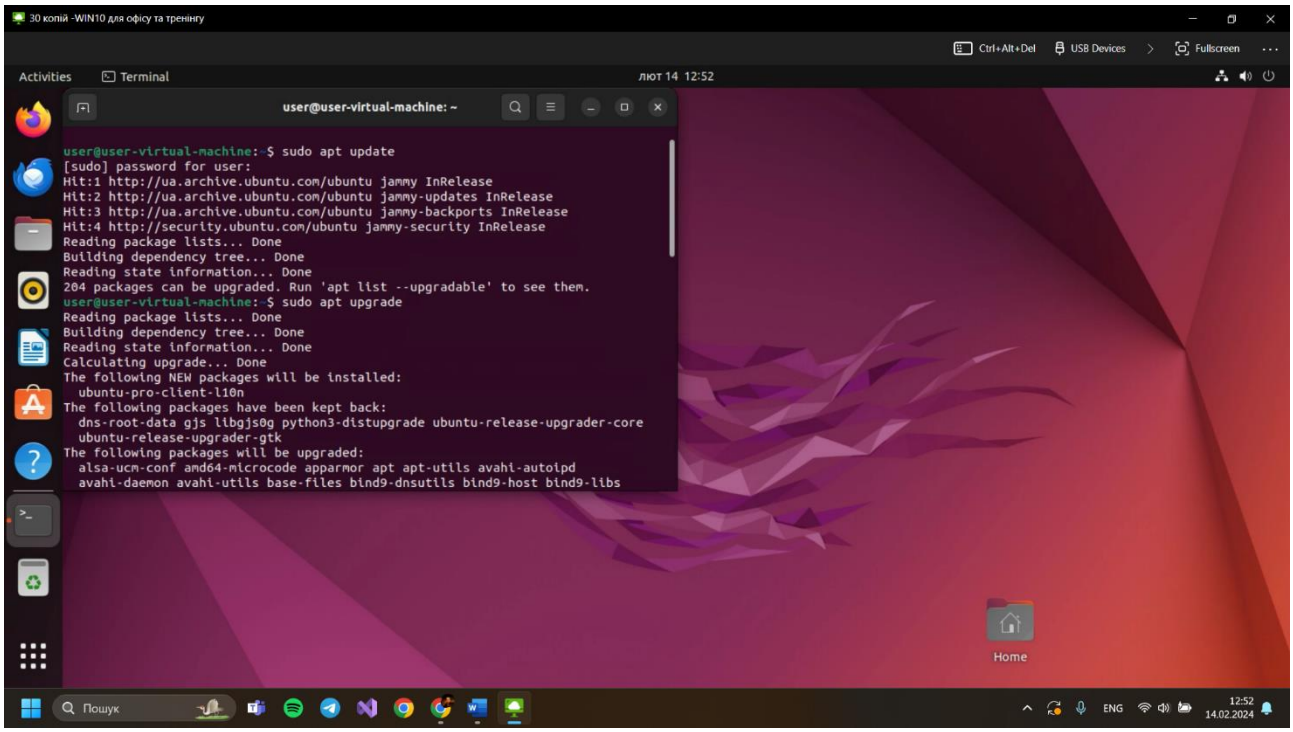






Входите в систему





Для коректної роботи вводите дві команди: **sudo apt update** та **sudo apt upgrade**

Контрольні питання

1. Що таке "віртуальна машина"? Призначення віртуальної машини.
2. Що означає GNU GPL?
3. Хто є автором ядра Linux?
4. Що таке ISO-образ?

Лабораторна робота №2. Файлові системи ОС Linux

Мета роботи: практичне знайомство з організацією даних основної файлової системи ОС Linux та утилітами, що використовуються.

1 КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Файл

Дані, що зберігаються на будь-якому носії, утворюють файл Linux. “Пристрої, підключені до комп'ютера (починаючи з клавіатури і закінчуючи будь-якими зовнішніми пристроями), Linux представляє як файли (так звані спеціальні файли). У Linux визначено кілька типів файлів. В основному користувач має справу з файлами трьох типів: звичайними файлами, призначеними для зберігання даних, каталогами та файлами-посиланнями.”[1]

Система файлів: каталоги

*Файлова система*має ієрархічну структуру. Linux може працювати з різними типами файлових систем. У цій роботі буде описано можливості файлової системи Ext3fs. У файловій системі Ext3fs кожен каталог - це окремий файл особливого типу ("d", від англ. "directory"), який відрізняється від звичайного файлу з даними: у ньому можуть міститися лише посилання інші файли і каталоги.

Допустимі імена файлів і каталогів

Linux завжди розрізняє великі та малі літери в іменах файлів і каталогів, тому "student", "Student" і "STUDENT" будуть трьома різними іменами.

Є спецсимволи "*", "\", "&", "<", ">", ";", "(", ")", "|", а також символи пробілу та табуляції.

Кодування та розширення

«У Linux в іменах файлів і каталогів можна використовувати не тільки символи латинського алфавіту, але і будь-які символи будь-якої мови.

У файловій системі Linux немає жодних розпоряджень щодо розширення. Визначити тип вмісту файлу можна і на основі самих даних (сигнатур). Багато форматів передбачають вказівку на початку файлу, як слід інтерпретувати подальшу інформацію.

У Linux є утиліта file, яка призначена для визначення типу даних, що містяться у файлі. «Ця утиліта ніколи не довіряє розширенню файлу (якщо воно є), а аналізує самі дані. file розрізняє як різні дані, а й різні типи файлів.

1.1 «Дерево каталогів

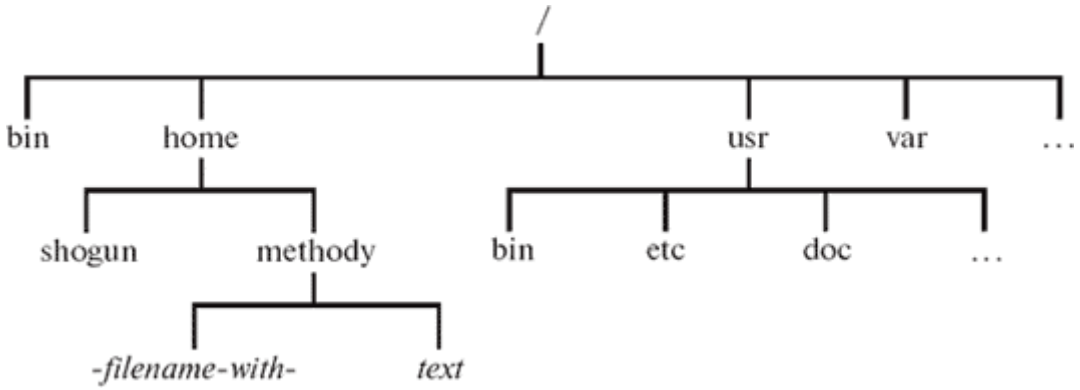
«У більшості сучасних файлових систем «використовується ієрархічна модель організації даних: існує один каталог, що поєднує всі

[1] https://linuxguide.rozh2sch.org.ua/

дані у файловій системі - це "корінь" усієї файлової системи, кореневий каталог. Кореневий каталог може містити будь-які об'єкти файлової системи, зокрема підкаталоги. Підкаталоги також можуть містити будь-які об'єкти файлової системи та підкаталоги і т. д. Таким чином, все, що записано на диску - файли, каталоги та спеціальні файли - обов'язково "належить" кореневому каталогу: або безпосередньо (міститься в ньому), або на деякому рівні вкладеності.

Структуру файлової системи можна уявити у вигляді дерева, коренем якого є кореневий каталог, а вершинах розташовані решта каталогів. На рис. 1 зображено дерево каталогів, курсивом позначені імена файлів, прямим зображенням - імена каталогів.

Рис. 1. Дерево каталогів в ext3fs



У будь-якій файловій системі Linux завжди є лише один кореневий каталог, який називається "/". Користувач Linux завжди працює з єдиним деревом каталогів, навіть якщо різні дані розташовані на різних носіях: жорстких» [23]або мережевих дисках, знімних дисках тощо. Яким є файлова система, використовується свій кореневий каталог, позначений літерою, наприклад "a", "c", "d" і т. д. Для того щоб відключати та підключати «файлові системи на різних пристроях до складу одного загального дерева, використовуються процедури монтування та розмонтування. Після того, як файлові системи на різних носіях підключені до загального дерева, дані, що містяться на них, доступні так, якби всі вони становили єдину файлову систему: користувач може навіть не знати, на якому пристрої які файли зберігаються.

Положення будь-якого каталогу в дереві каталогів описується за допомогою повного шляху. Повний шлях завжди починається від кореневого каталогу і складається з перерахування всіх вершин, що зустрілися під час руху по гілках дерева до каталогу, що шукається включно.» [4]Назви сусідніх вершин»[56] поділяються символом "/" ("слеш").»

[2] https://vipzone.net.ua/2023/06/01/1-вступ/
[3] https://linuxguide.rozh2sch.org.ua/
[4] https://vipzone.net.ua/2023/06/01/1-вступ/
[5] https://vipzone.net.ua/2023/06/01/1-вступ/
[6] https://linuxguide.rozh2sch.org.ua/

[7][8]У Linux повний шлях, наприклад, до каталогу "methody" у файловій системі, наведеній на рис. 1, записується наступним чином /home/methody.»[9]

1.2 Розміщення компонентів системи: стандарт FHS

Фрагмент дерева каталогів типової файлової системи Linux наведено на рис. 1. Утиліта ls виведе список всього, що міститься в цьому каталозі.

Приклад 1. Стандартні каталоги /. Використання утиліти ls

```
[ student@localhost ~]$ ls /
bin dev home lost+fount misc net proc tmp var
boot etc lib sbin usr
```

У прикладі 1 утиліта ls вивела список підкаталогів кореневого каталогу. Цей список буде приблизно таким самим у будь-якому дистрибутиві Linux. У кореневому каталозі Linux-системи зазвичай перебувають лише підкаталоги зі стандартними іменами. Короткий опис стандартної ієрархії каталогів Linux можна отримати за допомогою команди man hier. Повний текст та останню редакцію стандарту FHS можна прочитати за адресоюhttp://www.pathname.com/fhs/

«Вміст підкаталогів кореневого каталогу.

/bin Назва цього каталогу походить від слова "binaries" ("двійкові", "виконані"). У цьому каталозі знаходяться файли найнеобхідніших утиліт, які можуть знадобитися системному адміністратору або іншим користувачам.

/boot "Boot" – завантаження системи. У цьому каталозі знаходяться файли, необхідні для завантаження ядра - і, як правило, саме ядро. Користувачеві практично ніколи не потрібно безпосередньо працювати з цими файлами.

/dev знаходяться всі наявні в системі *файли* особливого типу, які використовуються для звернення до різних системних ресурсів та пристроїв. Наприклад, файли/dev/ttyN відповідають віртуальним консолям, деN- Номер віртуальної консолі. Дані, введені користувачем першої віртуальної консолі, система зчитує з файлу/dev/tty1;у цей файл записуються дані, які потрібно вивести користувачеві на цю консоль. У спеціальних файлах насправді не зберігаються жодні дані, за їх допомогою дані передаються.

---

[7] https://linuxguide.rozh2sch.org.ua/
https://vipzone.net.ua/2023/06/01/1-вступ/

[8] https://vipzone.net.ua/2023/06/01/1-вступ/

[9] https://linuxguide.rozh2sch.org.ua/

/ etc *«Каталог* для системних конфігураційних файлів Тут зберігається інформація про специфічні налаштування даної системи: інформація про зареєстрованих користувачів, доступні ресурси, налаштування різних програм.

/home Тут розташовані каталоги, що належать користувачам системи – домашні каталоги, звідси і назва "home". Відділення всіх файлів, створюваних користувачами, від інших системних файлів дає очевидну перевагу: серйозне пошкодження системи або необхідність оновлення не торкнеться файлів користувача.

/lib Назва цього каталогу - скорочення від "libraries" (англ. "Бібліотеки"). Щоб не включати ці функції до тексту кожної програми, використовуються стандартні функції бібліотек - це значно заощаджує місце на диску та спрощує написання програм. У цьому каталозі містяться бібліотеки, необхідні для роботи найважливіших системних утиліт, розміщених у
/bini/sbin.

/mnt *Каталог* для монтування (від англ. "mount") - тимчасового підключення файлових систем, наприклад, на знімних носіях (CD-ROM та ін.).

/proc У цьому вся каталозі всі файли " віртуальні " - вони розміщуються не так на диску, а оперативної пам'яті. У цих файлах міститься інформація про програми (процеси), що виконуються на даний момент у системі.

/root *Домашній каталог* адміністратора системи – користувача root. Сенс розміщувати його окремо від домашніх каталогів інших користувачів у тому, що/homeможе розміщуватися на окремому пристрої, який не завжди доступний (наприклад, на мережному диску), а домашній каталогrootповинен бути присутнім у будь-якій ситуації.

/sbin *Каталог* для найважливіших системних утиліт (назва каталогу -

скорочення від "system binaries"): на додаток до утиліт/binТут знаходяться програми, потрібні для завантаження, резервного копіювання, відновлення системи. Повноваження виконання цих програм є лише в системного адміністратора.

/tmp Цей каталог призначений для тимчасових файлів: у таких файлах програми зберігають необхідні для роботи проміжні дані. Після завершення роботи програми тимчасові файли втрачають сенс і мають бути видалені. Зазвичай каталог/tmpочищується при кожному завантаженні системи.

/usr Тут можна знайти такі ж підкаталогиbin,etc,lib,sbin, як і в кореневому каталозі Однак у кореневий каталог потрапляють лише утиліти, необхідні для завантаження та відновлення системи в аварійній ситуації - всі інші програми та дані розташовуються у підкаталогах/usr. Цей розділ файлової системи може бути великим.

/var Назва цього каталогу – скорочення від "variable" ("змінні" дані). Тут розміщуються ті дані, що створюються у процесі роботи різними програмами і призначені передачі іншим програмам і системам (черги друку, електронної пошти та інших.) чи відомості системного адміністратора (системні журнали, містять протоколи роботи системи). На відміну від каталогу/tmp сюди потрапляють ті дані, які можуть знадобитися після того, як їх створила
Програма завершила роботу.»[10][11]

Такий стандарт використовується у Linux-системах. Командна оболонка "знає", що файли, що виконуються, розташовуються в каталогах /bin, /usr/bin і т. д. - саме в цих каталогах вона шукає виконуваний файл cat. Завдяки цьому кожна знову встановлена в системі програма негайно виявляється доступною користувачеві з командного рядка. Для цього не потрібно ні перезавантажувати систему, ні запускати будь-які процедури - досить просто помістити файл, що виконується, в один з відповідних каталогів.

Рекомендації стандарту розміщення файлів і каталогів ґрунтуються на принципі розміщення файлів, які по-різному використовуються в системі, в різних підкаталогах. За типом використання файли можна розділити на такі групи: користувальницькі/системні файли

Файли користувача - це всі файли, створені користувачем і які не належать жодному з компонентів системи.

змінні/незмінні файли

До незмінених файлів відносяться всі статичні компоненти програмного забезпечення: бібліотеки, файли, що виконуються і т. д. - все, що не змінюється само без втручання системного адміністратора. Змінювані файли змінюються без втручання людини в процесі роботи системи: системні журнали, черги друку та ін. і дозволяє використовувати для зберігання цієї частини файлової системи CD-ROM та інші носії, доступні лише для читання.

файли, що розділяються/неподіляються

Це розмежування стає корисним, якщо йдеться про мережу, де

[10] https://linuxguide.rozh2sch.org.ua/

[11] https://vipzone.net.ua/2023/06/01/1-вступ/

працює кілька комп'ютерів. Значна частина інформації при цьому може зберігатися на одному з комп'ютерів і використовуватися іншими по мережі (до такої інформації відносяться, наприклад, багато програм і домашніх каталогів користувачів). Однак не можна розділяти частину файлів між системами (наприклад, файли для початкового завантаження системи).

Повний шлях до каталогу формально нічим не відрізняється від шляху до файлу, тобто на повному шляху не можна сказати напевно, є його останній елемент файлом або каталогом. Щоб відрізняти шлях до каталогу, іноді використовують запис із символом "/" в кінці шляху, наприклад, "/home/student/".

1.3 Поточний каталог

“Кожна виконувана програма "працює" в певному каталозі файлової системи. Такий каталог називається поточним каталогом. Можна уявляти, що програма під час роботи "перебуває" саме в цьому каталозі, це її "робоче місце". Залежно від поточного каталогу поведінка програми може змінюватися: найчастіше програма за замовчуванням працюватиме з файлами, розташованими саме в поточному каталозі - до них вона "дотягнеться" в першу чергу. Поточний каталог є у будь-якої програми, у тому числі і командної оболонки (shell) користувача. Оскільки користувач взаємодіє з системою через командну оболонку, можна говорити, що користувач "знаходиться" в тому каталозі, який в даний момент є поточним каталогом його командної оболонки.

Усі команди, що виконуються користувачем за допомогою shell, успадковують поточний каталог shell, тобто "працюють" у тому самому каталозі. Тому користувачеві важливо знати поточний каталог shell. Для цього служить утиліта pwd:

Команда pwd (print working directory) повертає повний шлях поточного каталогу командної оболонки – природно, саме тієї командної оболонки, за допомогою якої було виконано команду pwd.

Майже всі утиліти за замовчуванням читають та створюють файли у поточному каталозі. Наприклад, утилітаcat(concatenation – конкатенація) - виводить на екран вміст файлу "text":

[student@localhost student]$ cat text

Насправді командна оболонка, перш ніж передавати параметр "text" (ім'я файлу) утиліті cat, підставляє значення поточного каталогу - виходить повний шлях до файлу в файловій системі: "/home/student/text". Вміст даного файлу утиліта cat виведе на екран.”[12]

“Відносний шлях(Relative path) - шлях до об'єкта файлової системи, що не починається в кореневому каталозі. Для кожного процесу Linux визначено поточний каталог, з якого система починає відносний шлях під час виконання файлових операцій.

*Відносний шлях*будується так само, як і повний - перерахуванням

[12] http://ni.biz.ua/7/7_8/7_80548_tekushchiy-katalog.html

через "/всіх назв каталогів, що зустрілися при русі до шуканого каталогу або файлу. Між повним і відносним шляхом є тільки одна істотна відмінність: відносний шлях починається від поточного каталогу, в той час як повний шлях завжди починається від кореневого каталогу. Відносний шлях будь-якого файлу або каталогу у файловій системі може мати будь-яку конфігурацію - щоб дістатися до шуканого файлу, можна рухатися як у напрямку до кореневого каталогу, так і від нього. Linux розрізняє повний та відносний шлях дуже просто: якщо ім'я об'єкта починається на "/- це повний шлях, у будь-якому іншому випадку - відносний.

Відокремити шлях до файлу від імені можна за допомогою команд dirnameibasename відповідно."[13]

1.4 Домашній каталог

У Linux кожного користувача обов'язково є власний каталог, який і стає поточним відразу після реєстрації в системі - домашній каталог.

Домашній каталог(home directory) – це каталог, призначений для зберігання власних даних користувача Linux. Як правило, поточний безпосередньо після реєстрації користувача в системі. Повний шлях до домашнього каталогу зберігається в змінному оточенні HOME. Ім'я домашнього каталогу ~

Оскільки кожен користувач має в своєму розпорядженні власним каталогом і за замовчуванням працює у ньому, вирішується завдання поділу файлів різних користувачів. Зазвичай доступ інших користувачів до домашнього домашнього каталогу обмежений: найбільш типова ситуація, коли користувачі можуть читати вміст файлів один одного, але не мають права їх змінювати або видаляти.

1.5 Інформація про вміст каталогу – утиліта ls

Щоб мати можливість орієнтуватися у файловій системі, потрібно знати, що міститься у кожному каталозі. Переглянути вміст будь-якого каталогу можна за допомогою утиліти ls(Скорочення від англ. "list" - "список"):

Команда ls без параметрів виводить список файлів та каталогів, що містяться у поточному каталозі. Утиліта ls приймає один параметр – ім'я каталогу, вміст якого потрібно вивести. Ім'я може бути поставлене будь-яким доступним способом: у вигляді повного або відносного шляху.

Крім параметра, утиліта ls може використовувати безліч ключів, які потрібні для того, щоб виводити додаткову інформацію про файли каталогу або виводити список файлів вибірково. Щоб дізнатися про всі можливостіls, потрібно прочитати посібник з цієї утиліти за допомогою командиman ls.

Ключ –F використовується для того, щоб відрізняти файли від каталогів. За наявності цього ключа ls наприкінці імені кожного каталогу ставить символ "/", щоб показати, що він може утримуватися щось ще.

Утиліта ls за замовчуванням не виводить інформацію про об'єкти, чиє

[13] http://ni.biz.ua/7/7_8/7_80548_tekushchiy-katalog.html

ім'я починається з "." - у тому числі про "."і"..". Для того щоб переглянути повний список вмісту каталогу, і використовується ключ"-a(all). Як правило, з." починаються імена конфігураційних файлів та конфігураційних каталогів (на кшталт.bashrc), робота з якими (тобто налаштування оточення, "робочого місця") не перетинається з роботою над яким-небудь прикладним завданням.

Батьківський каталог(parent directory) - це каталог, у якому міститься цей. Для кореневого каталогу батьківським є він сам.

Посилання на поточний і батьківський каталог обов'язково присутні у кожному каталозі в Linux. Навіть якщо каталог порожній, тобто не містить жодного файлу чи підкаталогу, команда "ls -a" виведе список із двох імен: "."і"..". За посиланнями на поточний та батьківський каталоги можуть йти кілька файлів і каталогів, імена яких починаються з ".". Вони містяться налаштування командної оболонки (файли, що починаються з ".bash") та інших програм. У домашньому каталозі кожного користувача Linux завжди є кілька таких файлів. Використання цих файлів дозволяє користувачам незалежно один від одного налаштовувати поведінку командної оболонки та інших програм - організовувати своє робоче місце в системі.

1.6 Переміщення по дереву каталогів – команда cd

Користувач може працювати з файлами у своєму домашньому каталозі, а й у інших каталогах. В цьому випадку буде зручно змінити поточний каталог. Для зміни поточного каталогу командної оболонки використовується командacd(Від англ. "change directory" - "змінити каталог"). Командacdприймає один параметр: ім'я каталогу, куди потрібно переміститися - зробити поточним. Як ім'я каталогу можна використовувати повний або відносний шлях. У запрошенні командного рядка часто вказується поточний каталог shell – щоб користувачеві легше було орієнтуватися, в якому каталозі він "перебуває" в даний момент.

Командна оболонка вміє добудовувати імена файлів та каталогів: користувачеві достатньо набрати кілька перших символів імені файлу чи каталогу та натиснутиTab. Якщо є тільки один варіант завершення імені - оболонка закінчить його сама, і користувачеві не доведеться набирати символи, що залишилися. Добудовування - дуже суттєвий засіб економії зусиль та підвищення ефективності під час роботи з командним рядком. Сучасні командні оболонки можуть добудовувати імена файлів і каталогів, і навіть імена команд. Добудування найбільше розвинене в командному інтерпретаторіzsh. Оболонка PowerShell також може добудовувати імена.

Для переміщення до батьківського каталогу ("/home") зручно скористатися посиланням "..". Необхідність повернутися до домашнього каталогу з довільної точки файлової системи виникає досить часто, тому командна оболонка підтримує позначення домашнього каталогу за допомогою символу "~". Тому щоб перейти до домашнього каталогу з будь-якого іншого, достатньо виконати команду "cd ~". При виконанні команди символ "~буде замінено командною оболонкою на повний шлях до

домашнього каталогу користувача.

За допомогою символу "~" можна посилатися і на домашні каталоги інших користувачів: "~ім'я користувача". Командаcd, подана без параметрів, еквівалентна команді "cd ~і робить поточним каталогом домашній каталог користувача.

1.7 Створення каталогів – утиліта mkdir

У домашньому каталозі, як і будь-якому іншому, можна створювати скільки завгодно підкаталогів, у яких - свої підкаталоги тощо.

Щоб організувати таке піддерево, потрібно створити каталоги всередині домашнього. Для цього використовується утиліта mkdir. Вона застосовується з одним обов'язковим параметром: ім'ям каталогу, що створюється. За промовчанням каталог буде створено у поточному каталозі.

1.7.1 Створення нового порожнього файлу – команда touch

Для створення порожнього файлу з поточним часом створення використовується команда touch имя_нового_файла. Для вказівки дати створення у форматі ГГГГММДДhhmm використовується ключ –t. Наприклад

touch -t 0904080000 tst файл створено 8 квітня 2015р.

1.8 Копіювання та переміщення файлів

Для переміщення файлів та каталогів призначена утилітаmv(Від англ. "move" - "переміщати"). У mv два обов'язкові параметри: перший - файл або каталог, що переміщується, другий - файл або каталог призначення. Імена файлів і каталогів можуть бути задані у будь-якому допустимому вигляді: за допомогою повного чи відносного шляху. Крім того,mvдозволяє переміщати не лише один файл чи каталог, а одразу кілька. За подробицями про допустимі параметри та ключі слід звернутися до посібникаmv:

Переміщення файлу всередині однієї файлової системи насправді рівнозначне його перейменування: дані самого файлу при цьому залишаються на тих же секторах диска, а змінюються каталоги, в яких переміщення. Переміщення передбачає видалення посилання на файл з того каталогу, звідки він переміщений, і додавання посилання на цей файл у той каталог, куди він переміщений. В результаті змінюється повне ім'я файлу - повний шлях, тобто положення файлу у файловій системі.

Іноді потрібно створити копію файлу: для більшої безпеки даних, для того, щоб створити модифіковану версію файлу і т. п. У Linux для цього призначена утилітаср(Від англ. "Copy" - "Копіювати"). Утиліта ср вимагає використання двох обов'язкових параметрів: перший - копійований файл або каталог, другий - файл або каталог призначення. Як завжди, в іменах файлів та каталогів можна використовувати повні та відносні шляхи. Існує кілька варіантів комбінації файлів та каталогів у параметра хср- Про них можна прочитати в керівництві. Потрібно мати на увазі, що в Linux утиліта ср нерідко налаштована таким чином, що при спробі скопіювати файл поверх існуючого файлу ніякого попередження не виводиться. У цьому випадку файл буде просто перезаписано, а дані, які у старої версії файла,

безповоротно втрачені. Тому при використанні ср слід завжди бути уважним та перевіряти імена файлів, які потрібно скопіювати.

Створена за допомогою ср копія файлу пов'язана з оригіналом тільки у спогадах користувача, у файловій системі вихідний файл і його копія - дві абсолютно незалежні і нічим не пов'язані одиниці. Тому за наявності декількох копій одного і того ж файлу в рамках однієї файлової системи підвищується можливість заплутатися в копіях або забути про деякі з них. Якщо завдання полягає в тому, щоб забезпечити доступ до того самого файлу з різних точок файлової системи, потрібно використовувати спеціально призначений для цього механізм файлової системи Linux - посилання.

1.9 Файл та його імена: посилання

1.9.1 Жорсткі посилання – утиліта ln

Кожен файл являє собою область даних на жорсткому диску комп'ютера або іншому носії інформації, яку можна знайти на ім'я. У файловій системі Linux вміст файлу пов'язується з його ім'ям за допомогою жорстких посилань. Створення файлу за допомогою будь-якої програми означає, що буде створено жорстке посилання - ім'я файлу, і відкрито нову область даних на диску. Причому кількість посилань на ту саму область даних (файл) не обмежена, тобто у файлу може бути кілька імен.

Користувач Linux може додати файл ще одне ім'я (створити ще одне жорстке посилання на файл) за допомогою утиліти ln(Від англ. "link" - "з'єднувати, пов'язувати"). Перший параметр – це ім'я файлу, на який потрібно створити посилання, другий – ім'я нового посилання. За замовчуванням посилання буде створено у поточному каталозі:

Приклад 2. Створення твердих
посилань[student@localhost ~]$ ln text
text-hardlink

У прикладі 2 у домашньому каталозі користувача student створено жорстке посилання з ім'ям "text-hardlink"на файл"text". Якщо вивести докладний список файлів поточного каталогу та його підкаталогів ("ls - lR"), то у файлів "text"і"text-hardlink" Збігатимуться і розмір, і час створення. Тепер "text-hardlink"і"text- це два імені одного і того ж файлу.

Доступ до одного файлу за допомогою декількох імен може знадобитися в наступних випадках:

Одна й та сама програма відома під кількома іменами.

Доступ користувачів до деяких каталогів у системі може бути обмежений з міркувань безпеки. Однак якщо все ж таки потрібно організувати доступ користувачів до файлу, який знаходиться в такому *каталозі*, можна, можливо створити *жорстку посилання* на цей файл вдругом *каталозі*.

Сучасні файлові системи навіть на домашніх персональних комп'ютерах можуть налічувати до декількох десятків тисяч файлів та тисячі *каталогів*. Зазвичай у таких *файлових систем* складна багаторівнева

ієрархічна організація - в результаті шляху до багатьох файлів стають дуже довгими. Щоб організувати більш зручний доступ до файлу, який знаходиться дуже "глибоко" в ієрархії каталогів, можна використовувати жорстке посилання в більш доступному каталозі. Повне ім'я деяких програм може бути дуже довгим (наприклад,i586-alt-linux-gcc-3.3), до таким програмам зручніше звертатися за допомогою скороченого імені (*жорсткого посилання*) -gcc-3.3.

1.9.2 Індексні дескриптори

Оскільки завдяки жорстким посиланням файл може бути кілька імен, зрозуміло, що вся істотна інформація про файл у файловій системі прив'язана не до імені. У файлових системах Linux вся інформація, необхідна роботи з файлом, зберігається в індексному дескрипторі. Для кожного файлу існує індексний дескриптор: не тільки для звичайних файлів, але і для каталогів, файлів-дірок і т.д. Кожному файлу відповідає один індексний дескриптор.

Індексний дескриптор – це опис файлу, в якому міститься:

тип файлу (звичайний файл, каталог, спеціальний файл тощо);

*права доступу*до файлу;

інформація про те, кому належить файл;

позначки часу створення, модифікації, останнього доступу до файлу;

розмір файлу;

покажчики на фізичні блоки на диску, що належать цьому файлу - у цих блоках зберігається вміст файлу.

Всі індексні дескриптори пронумеровані, тому номер індексного дескриптора - це унікальний ідентифікатор файлу у файловій системі - на відміну від імені файлу (жорсткого посилання на нього), яких може бути кілька. Дізнатися номер індексного дескриптора будь-якого файлу можна за допомогою утиліти ls з ключем-і

Якщо вивести номери індексних дескрипторів файлу "text" і жорсткого посилання на нього "text-hardlink- можна побачити, що ці номери збігаються, тобто цим двом імен відповідає один індексний дескриптор, тобто один і той же файл.

Усі операції з файловою системою - створення, видалення та переміщення файлів - виробляються насправді над індексними *дескрипторами* а імена потрібні тільки для того, щоб користувач міг легко орієнтуватися у файловій системі. Більше того, ім'я (або імена) файлу в його індексному дескрипторі не вказано. У файловій системі Ext2 імена файлів зберігаються в каталогах: кожен каталог є списком імен файлів і номерів їх індексних дескрипторів. Жорстке посилання (ім'я файлу, що зберігається в каталозі) можна представляти як каталожну картку, де зазначений номер індексного дескриптора - ідентифікатор файла.

Жорстке посилання(hard link) - запис виду ім'я файла + номер індексного дескриптора в каталозі. Жорсткі посилання на Linux - основний

спосіб звернутися до файлу на ім'я.

1.9.3 Символьні посилання

У жорстких посилань є два суттєві обмеження:

*Жорстке посилання*може вказувати тільки на файл, але не на каталог, тому що в іншому випадку у файловій системі можуть виникнути цикли – нескінченні шляхи.

*Жорстке посилання*не може вказувати на файл в іншій файловій системі. Наприклад, на жорсткому диску неможливо створити жорстке посилання на файл, розташований на дискеті. Щоб уникнути цих обмежень, було розроблено символьні посилання. Посилання - це просто файл, в якому міститься ім'я іншого файлу. Символьні посилання, як і жорсткі, надають можливість звертатися до одного і того ж файлу за різними іменами. Крім того, символьні посилання можуть вказувати і каталог, чого не дозволяють жорсткі посилання. Символьні посилання називаються тому що містять символи - шлях до файлу чи каталогу.

Символьне посилання(symbolic link, файл-посилання) – це файл особливого типу ("l"), в якому міститься шлях до іншого файлу. Якщо на шляху до файлу зустрічається символове посилання, система виконує підстановку: вихідний шлях замінюється тим, що міститься в посиланні.
*Символьне посилання*можна створити за допомогою командиln з ключем "-s(скорочення від "symbolic").

Якщо виконати командуcat ім'я_файлу-посилання, то екран буде виведено вміст файлу, який вказує посилання.

Символьне посилання може містити ім'я неіснуючого файлу. У цьому випадку посилання буде існувати, але не буде "працювати": наприклад, якщо спробувати вивести вміст такого "битого" посилання за допомогою командиcat, буде видано повідомлення про помилку. Дізнатися, куди вказує символьне посилання можна за допомогою утиліти realpath.

1.10 Видалення файлів та каталогів – утиліти rm tarmdir

У ОС Linux для видалення файлів призначена утиліта rm (Скорочення від англ. "remove" - "видаляти"):

Якщо видалити файлtext у домашньому каталозі користувача student,файлtext-hardlink, який є жорстким посиланням на віддалений файлtext, збережеться, кількість жорстких посилань на цей файл зменшиться з "2"до"1- дійсно, text-hardlink- Тепер єдине ім'я цього файлу. Однак, якщо видалити і жорстке посилання text-hardlink, цей файл більше не залишиться жодного імені, він стане недоступним користувачеві і буде знищений.

Утиліта rm призначена для видалення жорстких посилань, а чи не самих файлів. У Linux, щоб повністю видалити файл, потрібно послідовно видалити всі жорсткі посилання на нього. При цьому всі жорсткі посилання на файл (його імена) рівноправні - серед них немає "головної", зі зникненням якої зникне файл. Поки є хоч одне посилання, файл продовжує

існувати. Втім, більшість файлів у Linux є лише одне ім'я (одне жорстке посилання на файл), тому команда rm Ім'я файлу в більшості випадків успішно видаляє файл.

Як мовилося раніше, символьні посилання - це окремі файли, тому після видалення файлу text, text-symlink, який посилався на цей файл, продовжує існувати, проте тепер це - "бите посилання", тому його також можна видалити командою rm.

Для видалення каталогів призначена інша утиліта -rmdir(Від англ. "Remove directory"). Втім, rmdir погодиться видалити каталог лише в тому випадку, якщо він порожній - у ньому немає жодних файлів та підкаталогів. Видалити каталог разом із усім його вмістом можна командоюrmз ключем "-r" (recursive). Командarm -r *каталог*- дуже зручний спосіб втратити відразу всі файли: вона рекурсивно обходить весь каталог, видаляючи все, що трапиться: файли, підкаталоги, символьні посилання ... а ключ "-f" (force) робить її роботу ще більш невідворотною, тому що придушує запити виду "видалити захищений від запису файл", так що rm працює безмовно і безупинно.

У Linux не передбачено процедури відновлення віддалених файлів та каталогів. Тому варто бути дуже уважним, віддаючи команду rm тим паче,rm -r: немає жодної гарантії, що видалені дані вдасться відновити.

1.11 Права доступу до файлової системи

1.11.1 Ідентифікатор користувача

Говорячи про права доступу користувача до файлів, зауважимо, що насправді маніпулює файлами не сам користувач, а запущений процес (наприклад, утиліта rm або cat). Оскільки і файл, і процес створюються і керуються системою, їй неважко організувати будь-яку політику доступу одних до інших, ґрунтуючись на будь-яких властивостях процесів.суб'єктівта файлів як об'єктів системи.

У Linux, однак, використовуються не будь-які властивості, а результат ідентифікації користувача – його UID. Кожен процес системи обов'язково належить якомусь користувачеві, і ідентифікатор користувача (UID) – обов'язкова властивість будь-якого процесу Linux. Коли програма login запускає стартовий командний інтерпретатор, вона приписує йому UID, отриманий у результаті діалогу. Звичайний запуск програми (exec()) або породження нового процесу (fork()) не змінюють UID процесу, тому всі процеси, запущені користувачем під час термінальної сесії, матимуть його ідентифікатор.

Оскільки UID однозначно визначається вхідним ім'ям, воно часто використовується замість ідентифікатора – для наочності. Наприклад, замість виразу "ідентифікатор користувача, що відповідає вхідному іменistudent", кажуть "UIDstudent(у наведеному нижче прикладі цей ідентифікатор дорівнює 500):

Приклад 3. Як дізнатися ідентифікатори користувача та членство у групах

```
[ student@localhost student]$ id
```

uid=500 (student) gid=500(student) групи=500 (student)

Утилітаidвиводить вхідне ім'я користувача та відповідний йому UID, а також групу за замовчуванням та повний список груп, членом яких він є.

1.11.2 Ідентифікатор групи

Користувач може бути членом декількох груп, так само як і кілька користувачів можуть бути членами однієї групи. Історично склалося так, що одна з груп – за замовчуванням – є для користувача основною - коли говорять про "GID користувача", мають на увазі саме ідентифікатор групи за замовчуванням. GID користувача вписаний в обліковий запис і зберігається в/etc/passwd, а інформація про відповідність імен груп їх ідентифікаторам, так само як і про те, до яких ще груп входить користувач – у файлі/etc/group. З цього випливає, що користувач не може не бути членом як мінімум однієї групи.

1.11.3 *Я*рлики об'єктів файлової системи

Під час створення об'єктів файлової системи – файлів, каталогів тощо. – кожному приписується ярлик. Ярлик включає UID – ідентифікатор користувача-господаря файлу, GID – ідентифікатор групи, якій належить файл, тип об'єкта і набір про атрибутів (код доступу), і навіть деяку додаткову інформацію. Атрибути (або код доступу) визначають, хто і що має право робити з файлом, описані нижче:

Приклад 4. Атрибути каталогів, показані командою ls -lразом 88

drwxr-xr-x	2 root root	4096 Квіт	4 2015 bin
drwxr-xr-x	4 root root	4096 Квіт	4 2016 boot
drwxr-xr-x	10 root root	3520 Квіт	5 14:26 dev
drwxr-xr-x	90 root root	8192 Квіт	5 14:22 etc
drwxr-xr-x	3 root root	4096 Квіт	4 21:22 home
drwxr-xr-x	11 root root	4096 Квіт	4 2016 lib
drwx------	2 root root	16384 Квіт	4 2016 lost+found
drwxr-xr-x	4 root root	4096 Квіт	5 14:22 media
drwxr-xr-x	2 root root	4096 Липень	11 2015 misc
drwxr-xr-x	2 root root	4096 жовт	20 2016 mnt
drwxr-xr-x	2 root root	0 Квіт	5 14:21 net
drwxr-xr-x	2 root root	4096 жовт	20 2016 opt
dr-xr-xr-x	106 root root	0 Квіт	5 2015 proc
drwxr-x---	31 root root	4096 Квіт	5 14:29 root
drwxr-xr-x	2 root root	8192 Квіт	4 2016 sbin
drwxr-xr-x	2 root root	4096 жовт	20 2016 selinux
drwxr-xr-x	2 root root	4096 жовт	20 2015 srv
drwxr-xr-x	11 root root	0 Квіт	5 2016 sys
drwxrwxrwt	16 root root	4096 Квіт	5 14:26 tmp
drwxr-xr-x	15 root root	4096 Квіт	4 2015 usr
drwxr-xr-x	21 root root	4096 Квіт	4 2015 var

Ключ -l утиліти ls визначає довгий (long) формат видачі (справа наліво): ім'я файлу, час останньої зміни файлу, розмір у байтах, група, господар, кількість жорстких посилань та рядок атрибутів. Перший символ у рядку атрибутів визначає тип файлу. Тип "-" відповідає " звичайному " файлу, а тип "d- каталогу (directory).

Незважаючи на те, що створення жорстких посилань на каталог неможливо, значення поля "кількість жорстких посилань" (другий стовпець) для всіх каталогів прикладу дорівнює двом, а не одному. Насправді цього і слід очікувати, тому що будь-який файловий каталог системи Linux завжди має не менше двох імен: власне (наприклад, tmp) та ім'я ".в самому цьому каталозі (tmp/.). Якщо в каталозі створити підкаталог, кількість жорстких посилань на цей каталог збільшиться на 1 за рахунок імені "..в підкаталозі (наприклад,tmp/subdir1/.):

1.11.4 Ієрархія прав доступу

Розглянемо докладніше, чому відповідають дев'ять символів у рядку атрибутів, що видаєтьсяls. Ці дев'ять символів мають вигляд .rwxrwxrwx" де деякі "r","w"i"x"можуть замінюватися на"-". Очевидно, літери відображають прийняті в Linux три види доступу - читання, запис та використання - проте в ярлику вони присутні у трьох екземплярах!"

“Справа в тому, що будь-який користувач (процес) Linux по відношенню до будь-якого файлу може виступати в трьох ролях: як господар (user), як член групи, якій належить файл (group), і як сторонній (other), жодних відносин власності на цей файл не має. Рядок атрибутів - це три трійки.rwx", що описує права доступу до файлу господаря цього файлу (перша трійка, "u"), групи, якій належить файл (друга трійка, "g") та сторонніх (третя трійка,"o"). Якщо в будь-якій трійці не вистачає літери, а замість неї стоїть"-", отже, користувачеві у відповідній ролі буде у відповідному вигляді доступу відмовлено.”[14]

При з'ясуванні відносин між файлом та користувачем, що запустив процес, роль визначається так:

Якщо UID файлу збігається з UID процесу, користувач – власник файлу

Якщо GID файлу збігається з GID будь-якої групи, до якої входить користувач, він є членом групи, якій належить файл.

Якщо ні UID, ні GID файлу не перетинаються з UID процесу та списком *груп*, в які входить користувач, що запустив його, цей користувач *сторонній*.

Саме ролі господаря користувач (процес) може змінювати ярлик файла. Єдине, чого не може робити господар зі своїм файлом – міняти йому господаря.

1.12 Використання прав доступу до Linux

1.12.1 Використання груп

У Linux визначено кілька системних груп, завдання яких – забезпечувати доступ членів цих груп до різноманітних ресурсів системи.

[14] https://learning.lpi.org/uk/learning-materials/010-160/5/5.3/5.3_01/

Часто такі групи носять назви, що говорять: "disk","audio","cdwriter" і т. п. Якщо звичайним користувачам доступ до деякого файлу, каталогу або спеціального файлу Linux закрито, він відкритий членам групи, якій цей об'єкт належить. Наприклад, у Linux майже завжди використовується віртуальна файлова система /proc– каталог, у якому як підкаталогів і файлів представлена інформація з таблиці процесів. Ім'я підкаталогу/procзбігається з PID відповідного процесу, а вміст цього підкаталогу відображає властивості процесу. Господарем такого підкаталогу буде господар процесу (з правами на читання та використання), тому будь-який користувач зможе переглянути інформацію про свої процеси. Саме каталогом/procкористується утилітаps. Використання утиліти ps для перегляду процесів Linux, що виконуються, розглядається в роботі “Процеси ОС Linux“.

1.13 Суперкористувач

Суперкористувач- єдиний користувач Linux, на якого не поширюються обмеження прав доступу. Має нульовий ідентифікатор користувача.

Суперкористувач Linux – це виділений користувач системи, на якого не поширюються обмеження прав доступу. UID суперкористувацьких процесів дорівнює0: так система відрізняє їх від інших користувачів. Саме суперкористувач має можливість довільно змінювати власника та групу файлу. Йому відкритий доступ на читання та запис до будь-якого файлу системи та доступ на читання, запис та використання до будь-якого каталогу. Нарешті, суперкористувацький процес може на якийсь час змінити свій власний UID з нульового на будь-який інший. Саме так і вчиняє програмаlogin, коли, провівши процедуру ідентифікації користувача, починає стартовий командний інтерпретатор.

Серед облікових записів Linux завжди є запис на ім'я root ("корінь"), що відповідає нульовому ідентифікатору, тому замість "суперкористувач" часто говорять "root". Багато системних файлів належатьroot, безліч файлів тільки йому доступні для читання або запису. Пароль цього облікового запису – одна з найбільших коштовностей системи. Саме за її допомогою системні адміністратори виконують найвідповідальнішу роботу.

Існує два різні способи отримати права суперкористувача. Перший – це зареєструватися в системі під цим ім'ям, ввести пароль та отримати стартову оболонку, що має нульовий UID. Це – найнеправильніший спосіб, користуватися яким варто, тільки якщо не можна застосувати інші. В Ubuntu описаний спосіб не використовується, замість нього використовується другий спосіб.

Другий спосіб - скористатися спеціальною утилітою sudo, яка дозволяє виконати одну або кілька команд від імені іншого користувача. За умовчанням ця утиліта виконує команду від імені користувача root, тобто запускає командний інтерпретатор із нульовим UID. Відмінність від попереднього способу полягає в тому, що завжди відомо, хто саме запускав

sudo, Отже, зрозуміло, з кого запитувати за наслідки.

1.14 Пошук файлів

Для пошуку файлу на ім'я або його частину використовується утиліта locate. Параметр визначає ім'я файлу. Для пошуку без урахування регістру є ключ –i.

Для обмеження обсягу інформації, що виводиться, використовується ключ -n Число. Порядковий висновок виходить, якщо результати пошуку направити конвеєром в програму less, наприклад locate mp3 | less

Утиліта locate веде пошук у базі даних, яка має періодично оновлюватися утилітою updatedb, що виконується з правами адміністратора. Інший спосіб знайти файл надає утиліта find. Її ключі наведені у (табл. 1).

Таблиця 1.

Ключі утиліти find

Ключ	Призначення
-name	Вказує ім'я файлу або його частину
-size	Задає розмір файлу, наприклад 12k
-type	Задає тип об'єкта для пошуку: f-звичайний файл d-каталог l-символьне посилання
-a	Логічна зв'язка and
-o	Логічна зв'язка or
-user	Вказує ім'я користувача

Перевагами утиліти find є незалежність від бази даних та широкі функціональні можливості, недолік – менша швидкість пошуку порівняно з locate.

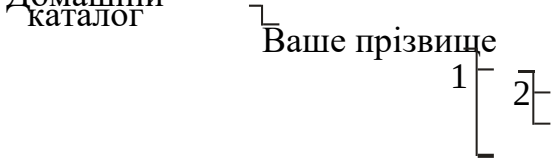
2 Хід виконання

1. Ознайомитись з теоретичними відомостями.
2. Після завантаження ОС Linux та запиту імені ввести ім'я та пароль користувача.
3. Після завантаження ОС запустити термінал.

Усі завдання роботи слід виконати у режимі командного рядка за допомогою терміналу.

Завдання:

1. Створити у домашньому каталозі наступну структуру підкаталогів (існуючі каталоги не видаляти!):



3
4

2. Скопіювати файл /etc/group у каталоги 1, 2, 3 і 4 використовуючи абсолютні імена копіюваного файлу та каталогу призначення.
3. За допомогою утиліти file вивести на екран відомості про 3 - 4 різні файли (у тому числі з каталогів /bin та /dev).
4. Виконати команду ls –l /dev використовуючи таблицю 2 позначень типів файлів, перерахувати типи файлів, що зберігаються в каталозі /dev

Таблиця 2.

Позначення типів файлів

Символ	Тип файлу
d	Каталог
l	Символьне посилання
s	Сокет
b	Блоковий пристрій
c	Символьний пристрій
p	Іменований канал

5. Використовуючи довідкову систему, ознайомитись із ключами утиліти ls -R, -1 (одиниця), -m, --color, ключі, що визначають порядок виведення на екран
6. Створити жорстке та символічне посилання для одного із створених у п.2 файлів.

Таблиця 3.

Індивідуальні завдання

варіант	Завдання
1	Вивести список імен файлів із /var, використовуючи ключ –l Список упорядкувати за розмірами файлів. 2. Знайти файли, імена яких закінчуються на PDF
2	Вивести список імен файлів із /bin, використовуючи ключ –l Список упорядкувати за датами створення
	2. Знайти файли, імена яких закінчуються на jpg
3	Вивести список імен файлів із /sbin, використовуючи ключ –l Список впорядкувати за іменами 2. Знайти файли, розміри яких перевищують 25к (запис+25k)
4	Вивести список імен файлів з /tmp, використовуючи ключ –l Список впорядкувати за іменами 2. Знайти файли, імена яких закінчуються на text

5	Вивести список імен файлів із /usr, використовуючи ключ –l Список упорядкувати за розмірами файлів. 2. Знайти файли, імена яких закінчуються на jpg та розміри більше 1к
6	Вивести список імен файлів із /bin, використовуючи ключ –l Список впорядкувати за датами створення 2. Знайти файли, розміри яких перевищують 15к (запис+15k)
7	Вивести список імен файлів із /usr, використовуючи ключ –l Список упорядкувати за розмірами файлів. 2. Знайти файли, розміри яких перевищують 25к (запис +25k) та імена починаються на s
8	Вивести список імен файлів з /var, використовуючи ключ –l Список впорядкувати за датами створення 2. Знайти файли, розміри яких перевищують 25к (запис +25k) та імена починаються на s, а закінчуються на jpg
9	Вивести список імен файлів із /sbin, використовуючи ключ –l Список впорядкувати за розмірами файлів 2. Знайти файли, розміри яких перевищують 1М (запис +1m)
10	Вивести список імен файлів із /bin, використовуючи ключ –l Список впорядкувати за іменами 2. Знайти файли, розміри яких перевищують 5к (запис +5k)

7. Результати оформити у звіті.

3 Звіт про роботу

Зміст звіту має включати завдання, опис команди на вирішення завдання, синтаксис команди, результати роботи команди. На завершення звіт доповісти включати відповіді на контрольні питання

Увага! Опис ваших дій має підкріплюватися скріншотами.

4 Контрольне питання

1. Монтування та демонтування файлової системи в Linux

Лабораторна робота №3. “Система розмежування доступу в UNIX, права доступу до файлів”

Мета: здобути навички роботи з системою рожмежування доступу
Завдання до виконання:

1. Створіть каталог lr3.
2. Скопіюйте в каталог lr3 файл /bin/cat під назвоюmy_cat
3. За допомогою файлу my_cat, який знаходиться в каталозі lr3, перегляньте файл .profile
4. Ознайомтесь зі списком файлів у каталозі lr3. Потім перегляньте всі файли з повною інформацією (права доступу, власник, дата модифікації файлу). Продивіться оригінальний файл і порівняйте.
5. Змініть права доступу до файлу my_cat на виключно читання.
6. Повторити завдання із пункту 3.
7. Змініть права на файл my_cat таким чином, щоб Ви могли вносити зміни у файл, а всі інші - ні.
8. Перегляньте простий список файлів у цьому каталозі. Запустіть і видаліть файл my_cat з каталогу.
9. Прокоментуйте результати.
10. Використовуючи команду su <user name>, авторизуйтесь в системі, використовуючи обліковий запис іншого користувача. Створіть каталог lr3_2.
11. Знову аторизуйтесь в системі, користуючись своїм обліковим записом. Зробіть власником каталогу lr3 іншого користувача. Зробіть себе власником каталогу lr3_2. Прокоментуйте.
12. Зайдіть у каталог lr3. Присвойте файлам і каталогам права доступу відповідно до таблиці варіантів. Створіть новий файл і каталог і перевірте налаштування.
13. Поверніть собі права редагування каталогу.
14. Створіть у каталозі lr3 каталог acl_test та у ньому файли file1, file2. Під час створення file1 командою echo додайте до нього довільний текст.
15. Виведіть ACLдля file1
16. Внесіть зміни до прав доступу до file1 таким чином, щоб лише власник мав право на читання.

Лабораторна робота 4. Редактор vi

Мета: Оволодіння практичними навичками роботи з редактором vi

Завдання до виконання

1. Завантажтеся в систему під Вашим користувацьким ім'ям.
2. Використовуючи редактор vi необхідно скласти текстовий файл text. Файл має містити текст. У тексті має бути ваше прізвище. Збережіть нові файли з іменами text та text1, вийдіть із редактора.
3. Установіть на файл text1 права доступу так, щоб Ви могли тільки читати цей файл, але не модифікувати його.
4. Завантажте файл text у редактор, скопіюйте перші 2 рядка тексту в буфер і вставте їх у кінець тексту.
5. Запишіть файл під тим же ім'ям.
6. Не виходячи з редактора, завантажте файл text1 і, попередньовідкривши новий shell і змінивши права доступу на файл, запишіть файл, не виходячи з редактора.
7. Користуючись пойменованими буферами, перенесіть 3 рядки тексту з першого файлу в другий. Збережіть зміни. Вийдіть з редактора.
8. Відкрийте у редакторі файл text і в кінець його додайте уміст файлу text1.
9. Запишіть отриманий файл як text2 і, не виходячи з редактора, видаліть файли texti text1.

Лабораторна робота 5. Командна оболонка shell, стандартні потоки вводу/виводу, фільтри і конвеєри

Мета: реплізувати роботу з фільтрами та конвеєрами, ознайомитись з командною оболонкою.

Завдання до виконання

1. Зайдіть у каталог “**/bin”**. Продивіться список усіх файлів, щопочинаються із символу, відповідно до варіанту.
2. Перегляньте список файлів, імена яких складаються з певної кількості символів відповідно до варіанту.
3. Створіть в Вашому домашньому каталозі підкаталог lr5 і перейдіть.
4. Створіть новий файл my_text і запишіть у нього текст. Потім допишіть ще кілька рядків.
5. Підрахуйте кількість файлів у каталозі з використанням конвеєрів та без них. Порівняйте результат.
6. Підрахуйте кількість файлів у каталозі, при чому зберігайте список файлів у файлі filelist, використовуючи відповідну команду.
7. Виведіть на екран назви усіх файлів і каталогів, що починаються з літери “_m”.
8. Починаючи з кореневого каталогу, виведіть на екран імена всіх каталогів, що останній раз змінювалися більш 15 днів назад.
9. Виведіть на екран час, що реалізується командою “data”.
10. Виведіть на екран імена усіх файлів у каталозі /bin, що містять слова Software чи software. Помилки не повинні виводитися на екран.

Лабораторна робота 6. Процеси та їх створення в Win32 API для ОС MS Windows

Мета: навчитися керувати роботою процесу в ОС MS Windows

«Windows під процесом розуміється об'єкт ядра, якому належать системні ресурси, використовувані додатком. Тому можна сказати, що у Windows процесом є програма. Виконання кожного процесу починається з первинного потоку, який передбачає виділення процесорного часу та ресурсів. У процесі свого виконання процес може створювати й інші потоки. Виконання процесу закінчується після завершення роботи всіх його потоків. Процес може бути завершений викликом функцій ExitProcess і TerminateProcess.[15]»

Новий процес у Windows створюється викликом функції CreateProcess, яка має наступний прототип:

BOOLCreateProcess (LPCTSTRlpApplicationName, // ім'я модуля, що виконується.

LPTSTRlpCommandLine, // командний рядок

LPSECURITY_ATTRIBUTES lpProcessAttributes, //атрибути захисту процесу LPSECURITY_ATTRIBUTES lpThreadAttributes, // атрибути захисту потоку

BOOL bInheritHandle, // ознака успадкування дескриптора

DWORD dwCreationFlags, // прапори створення процесу

LPVOIDlpEnvironment, // блок нового середовища оточення

LPCTSTRlpCurrentDirectory, // поточний каталог

LPSTARTUPINFOlpStartUpInfo // вид головного вікна

LPPROCES S_INFORMATION lpProcessInformation// інформація про процес);

Функція CreateProcess породжує новий дочірній процес та його перший потік (нитка). У цьому процесі виконується вказаний файл lpApplicationNamez командним рядкомlpCommandLine. Втім, параметр lpApplicationName може бути дорівнює nil (порожній рядку), а ім'я виконуваного модуля в цьому випадку має бути першим елементом командного рядка, що задається параметром lpCommandLine. Сам модуль може бути будь-якого виду: 32-розрядним додатком Windows, додатком MS-DOS, OS/2 і т.п. Однак, якщо з програми Windows створюється процес MS-DOS, то параметр lpApplicationName повинен бути рівним (пустому рядку), а ім'я файлу та його командний рядок включаються в lpCommandLine. Так що, як правило, щоб не помилитися, простіше завжди задавати lpApplicationName = nil і поміщати всю інформацію в lpCommandLine.

Наприклад розглянемо таку програму. Запустимо програму Notepad.exe. ми передаємо системі ім'я нового процесу та його параметри через командний рядок. У цьому випадку ім'я нового процесу може і не містити повного шляху до exe-файлу, а лише ім'я самого exe-файлу. У разі використання параметра lpCommandLine система для запуску нового процесу здійснює пошук необхідного exe-файлу в послідовності каталогів. Програма, яка запускає блокнот із командного рядка, виглядає так.

Програма 1.

[15] https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/34310/90424.pdf?sequence=2

// Приклад запуску процеса Notepad

```
#include"stdafx.h"
#include <windows.h>
 #include <iostream>
using namespace std;
void main()
{
STARTUPINFO si;
PROCESS_INFORMATION pi;
ZeroMemory(&si, sizeof(STARTUPINFO));
si.cb = sizeof(STARTUPINFO);
wchar_t pCmdLine []= (L"Notepad.exe");
// завантажуєм процес Notepad
if (!CreateProcess(NULL, pCmdLine, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
{
cout << "The new process is not created." << endl
      << "Check a name of the process." << endl;
cin.get();
cin.get();
}
Sleep(1000);
cout<<" The new process is created. "<<endl;
CloseHandle (pi.hThread);
CloseHandle (pi.hProcess);
cin.get();
cin.get();
}
```

Завершення процесів.

Процес може завершити свою роботу викликом функції ExitProcess, яка має такий прототип:

VOIDExitProcess(UINTuExitCode// код возврата для всіх потоков);

Наприклад:

ExitProcess(1);

Один процес може завершити інший процес за допомогою виклику функції TerminateProcess, яка має наступний прототип:

```
BOOL TerminateProcess(
HANDLE hProcess,
UINT uExitCode);
```

Якщо функція TerminateProcess виконалася успішно, вона повертає значення TRUE. Інакше повертається значення FALSE. Функція TerminateProcess завершує роботу процесу, але не звільняє всі ресурси цього процесу. Тому ця функція повинна викликатись лише в аварійних ситуаціях при зависанні процесу.

Завдання до виконання

1.Ознайомтеся з теоретичними відомостями.

2.Розберіть програму 1. Запустіть програму виконання в середовищі MSVisualStudio. Опишіть результат виконання програми у звіті.

3.Напишіть програму 2, яка б виводила на консоль Ваше ім'я, прізвище, курс, групу. Надайте progXXX.cpp, де XXX - Ваше прізвище.

4. Відкомпілюйте цю програму та отримайте файл progXXX.exe.

5.Напишіть програму 3, яка запускатиме програму progXXX.exe як новий процес.

6. До файлу звіту додати заархівовані проєкти програм.

Лабораторна робота 7. Робота з віртуальною пам'яттю в ОС MS Windows

Мета: навчитися управляти віртуальною пам'яттю в ОС MSWindows

Теоретичні відомості

«Для резервування або розподілу області віртуальної пам'яті процес має викликати функцію VirtualAlloc, яка має наступний прототип:

LFVOIDVirtualAlloc(

LPVOIDlpAddress // область для розподілу або резервування

SIZE_TdwSize, // розмір області

DWORDflAllocationType, // тип розподілу DWORDflProtect// тип захисту доступу

);

У разі успішного завершення ця функція повертає адресу віртуальної пам'яті, розподіленої або зарезервованої процесом, а у разі невдачі - null. При цьому відзначимо таку деталь, якщо розподіл віртуальної пам'яті функцією VirtualAlloc завершується успішно, виділена пам'ять автоматично ініціалізується нулями. Опишемо призначення параметрів цієї функції.

Параметр lpAddress встановлюється програмою, що викликає, і вказує системі початкову адресу віртуальної пам'яті, яку процес хоче зарезервувати або розподілити. Ця адреса може вказувати як на вільну, так і на зарезервовану раніше віртуальну пам'ять. При встановленні цієї адреси слід розрізняти такі ситуації:

- у разі резервування віртуальної пам'яті ця адреса вирівнюється системою до межі 64 Кбайт, яка передує зазначеній адресі;
- у разі розподілу віртуальної пам'яті із зарезервованої раніше області ця адреса округляється операційною системою до межі віртуальної сторінки, що містить цю адресу;
- у разі, якщо параметр lpAddress дорівнюєnull, то операційна система сама вибирає початкову адресу області віртуальної пам'яті.

Параметр dwsize встановлюється програмою, що викликає, і вказує розмір розподіленої або резервованої області віртуальної пам'яті в байтах. Якщо параметр lpAddress дорівнює NULL, то система округляє цю величину у бік до кратності розміру віртуальної сторінки. Якщо ж пам'ять розподіляється за конкретними адресами, то ця пам'ять включатиме всі сторінки, які містять байти з діапазону від lpAddress до lpAddress+dwSize.

Параметр flAllocationTуревстановлюється програмою та вказує на тип операції, яку виконує функція VirtualAlloc.Значенням цього параметра може бути будь-яка комбінація наступних прапорів:

- mem _commit - розподілити пам'ять програмі;
- mem_reserve - зарезервувати область фізичної пам'яті.

Параметр flProtect встановлює атрибути доступу до віртуальної пам'яті, які дозволяють виконувати над сторінками віртуальної пам'яті певні операції. Цей

параметр може бути комбінацією наступних прапорів:»[16]

· PAGE_READWRITE - Читання та запис.

- PAGE_READONLY - Тільки читання.
- PAGE_EXECUTE - Тільки виконання програмного коду.
- PAGE_EXECUTE_READ - Виконання та читання.
- PAGE_EXECUTE_READWRITE - Виконання, читання та запис.
- B· PAGE_NOACCESS - Заборонено будь-який вид доступу.
- PAGE_GUARD - Сигналізація доступу до сторінки. Це значення можна використовувати разом із будь-якими іншими, крім PAGE_NOACCESS.

Потім необхідно звільнити віртуальну пам'ять з використанням функції VirtualFree:

```
BOOL VirtualFree
(
        LPVOID lpAddress, // адреса пам'яті
        SIZE_T dwSize, // розмір пам'яті
        DWORD dwFreeType // Операція
);
```

При успішному виконанні функція поверне ненульове значення, а разі невдачі - false. Параметри цієї функції можуть приймати такі значення:

- dwSize - розмір (в байтах), якщо ми будемо використовувати тип звільнення, як MEM_RELEASE, то розмір має бути встановлений 0.

- dwFreeType - визначатиме яка операція відбудеться з пам'яттю:

B· MEM_RELEASE - звільнена

· MEM_DECOMMIT - зарезервовані, але не використовувані.

Якщо в параметрі dwFreeType буде встановлено значення MEM_RELEASE, то ця адреса має співпадати з адресою, яку повернула функція VirtualAlloc.

Розглянемо програму, яка розподіляє область віртуальної пам'яті під одновимірний масив цілих чисел, та був звільняє її. Зазначимо один момент у цій програмі. Щоб відразу розподілити область віртуальної пам'яті, ми повинні встановити параметр lpAddressв значенняnull.В цьому випадку система сама визначає початкову адресу області віртуальної пам'яті для розподілу процесу.

Програма1. Розподіл віртуальної пам'яті процесу

```
#include<windows.h>
#include<iostream>
Використовуючи назву std;
int main()
 {
int *a;
```

[16] https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/34310/90424.pdf?sequence=2

```
constint size=1000;
a = (int *) VirtualAlloc (
NULL,
size * sizeof (int), MEM_COMMIT, PAGE_READWRITE);
if (!a)
 {
cout << "Virtual allocation failed." << endl;
return GetLastError();
  }
cout << "Virtual memory address: " << a << endl;
// звільняємо віртуальну пам'ять
if (!VirtualFree (a, 0, MEM_RELEASE))
{
cout << "Memory release failed." << endl;
return GetLastError();
}
return 0;
}
```

Завдання до виконання:

1.Ознайомтеся з теоретичними відомостями.

2.Розберіть програму 1. Запустіть програму виконання в середовищі MSVisualStudio. Опишіть результат виконання програми у звіті. Щоб переглянути результати виконання програми, вставте функції затримки екрана консолі, куди виводяться повідомлення програми.

3.Напишіть програму 2, яка розподіляє область віртуальної пам'яті під двомірний масив цілих чисел. Надайте progXXX.cpp, де XXX - Ваше прізвище.

4. Відкомпілюйте цю програму та отримайте файл progXXX.exe.

5. До файлу звіту додати заархівовані проєкти програм.

Лабораторна робота №8. Створення проекту бібліотеки динамічного компонування (DLL) з прикладу використання MS Visual Studio 2013.

1. У рядку меню виберіть Файл, Створити, Проект.
2. У лівій області діалогового вікна Створити проект розгорніть Встановлені, Шаблони, Visual C++ і виберіть Win32.
3. У центральній області виберіть Консольну програму Win32.
4. Введіть ім'я для проекту, наприклад MyDll, у полі Ім'я. Вкажіть ім'я для вирішення, наприклад MyDll, у полі Ім'я рішення. Натисніть кнопку OK.
5. На сторінці Огляд діалогового вікна Майстер програм Win32 натисніть кнопку Далі.
6. На сторінці Параметри програми виберіть у полі Тип програми пункт бібліотека DLL.
7. Натисніть кнопку Готово, щоб створити проект.

Додавання класу до бібліотеки динамічного компонування

1. Щоб створити файл заголовка для нового класу, у меню Проект виберіть пункт Додати новий елемент. У діалоговому вікні Додати новий елемент у лівій області у розділі Visual C++ виберіть Код. У центральній області виберіть Заголовний файл (.h). Вкажіть ім'я файлу заголовка, наприклад MyDll.h, а потім натисніть кнопку Додати. Показано порожній заголовковий файл.
2. Додайте наступний код на початок файлу заголовка:

```
// MyDll.h.
#ifdefMYDLL_EXPORTS
#defineMYDLL_API __declspec(dllexport)
#else
#defineMYDLL_API __declspec(dllimport)
#endif
```

3. Додати базовий клас з ім'ям My_class для виконання загальних математичних операцій, таких як складання та віднімання. Код повинен виглядати приблизно так:

```
namespace MathFuncs
{
// Цей клас експортується з MyDLL.dll
class My_class
{
public:
// a + b
static MyDll_API double Add(double a, double b);
// a - b
```

```
static MyDll_API double Subtract(double a, double b);
};
}
```

4. У проекті MyDll у браузері рішень у папці Вихідні файли відкрийте файл MyDll.cpp.
5. Реалізуйте функціональність класу My_class у вихідному файлі. Код повинен виглядати приблизно так:

```
#include "stdafx.h"
#include "MyDll.h"
#include <stdexcept>
using namespace std;
namespace MathFuncs
{
double My_class::Add(double a, double b)
{
return a+b;
}
double My_class::Subtract(double a, double b)
{
return a - b;
}
}
```

6. Скомпілюйте бібліотеку динамічного компонування, вибравши Зібрати рішення в меню Складання.

Створення програми, яка посилається на DLL

1. Щоб створити програму C++, яка буде посилатися і використовувати створену раніше бібліотеку DLL, в меню Файл виберіть Створити і потім Проект.
2. У лівій області категорії Visual C++ виберіть Win32.
3. У центральній області виберіть Консольну програму Win32.
4. Введіть ім'я проекту, наприклад MyExecDll, у полі Ім'я. У списку, що розкривається, поруч із полем Рішення виберіть Додати до рішення. В результаті новий проект буде додано до рішення, що містить бібліотеку DLL. Натисніть кнопку OK.
5. На сторінці Огляд діалогового вікна Майстер програм Win32 натисніть кнопку Далі.
6. На сторінці Параметри програми виберіть у полі Тип програми пункт Консольна програма.
7. На сторінці Параметри програми у розділі Додаткові параметри зніміть прапорець Попередньо скомпільований заголовок.
8. Натисніть кнопку Готово, щоб створити проект.

Використання функціональності з бібліотеки класів у додатку

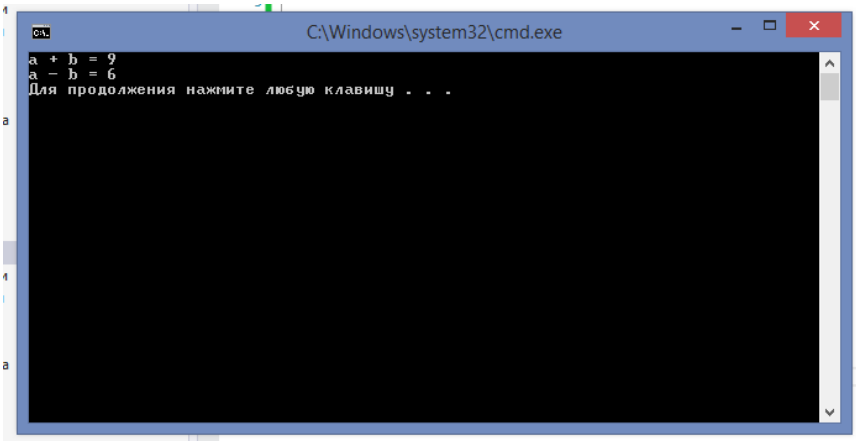
1. Після створення консольної програми буде створено порожню програму. Ім'я вихідного файлу співпадатиме з раніше вибраним ім'ям. У цьому прикладі має ім'я MyExecDll.cpp.
2. Для використання у програмі математичних процедур, створених у бібліотеці DLL, необхідно послатися на цю бібліотеку.Для цього в браузері рішень виберіть проект MyExecDll, а потім у меню Проект виберіть пункт Посилання. У діалоговому вікні Сторінки властивостей розгорніть вузол Загальні властивості, виберіть посилання та натисніть Додати нове посилання.
3. У діалоговому вікні Додати посилання перелічені бібліотеки, на які можна створити посилання. На вкладці Проект перераховані всі проекти поточного рішення та включені до них бібліотеки, якщо вони є. Встановіть прапорець поруч із MyDll на вкладці Проекти, а потім натисніть кнопку ОК.
4. Щоб створити посилання на файли заголовка DLL, необхідно змінити шлях до каталогів включення. Для цього в діалоговому вікні Сторінки властивостей послідовно розгорніть вузли Властивості конфігурації та C/C++, а потім виберіть Загальні. У полі Додаткові каталоги файлів вкажіть шлях до місця розміщення файлу заголовка MyDll.h. Потім натисніть кнопку ОК.
5. Тепер класMy_classможна використовувати у додатку. Замініть вміст файлу MyExecDll.cpp на наступний код.

```
#include <iostream>
#include "MyDll.h"
using namespace std;
int main()
{
double a = 7.5;
double b = 1.5;
cout << "a + b = " <<
MathFuncs:: My_class::Add(a, b) << endl;
cout << "a - b = " <<
MathFuncs:: My_class::Subtract(a, b) << endl;
return 0;
}
```

Зберіть файл, що виконується, вибравши команду Зібрати рішення в меню Складання.

Запуск програми

1. Переконайтеся, що проект MyExecDll вибраний як проект за промовчанням. У браузері рішень виберіть MyExecDll і потім у меню Проект виберіть Призначити запускаемим проектом.
2. Щоб запустити проект, у рядку меню виберіть Налагодження, Запуск без налагодження. Результат виконання повинен виглядати так:



Завдання.

1. Відпрацюйте приклад створення динамічної бібліотеки та програми, яка використовує цю бібліотеку. Отриманий результат прикріпіть до звіту.
2. Створіть динамічну бібліотеку, яка виводила б на консоль Ваше ім'я та прізвище. Розробіть програму, яка використовувала б цю бібліотеку.

Список використаних джерел

1. Stallings W. Operating Systems. Internals and Design Principles / William Stallings. – Ninth Edition. – Pearson Education Limited, 2018. – 1128 p.
2. Tanenbaum A. Modern Operating Systems / Andrew S. Tanenbaum, Herbert Bos. – 4th Edition. – Pearson Education, Inc., 2015. – 1137 p.
3. Голубничий Д.Ю. Операційні системи [Електронний ресурс]/ Д.Ю.Голубничий, А.В. Холодкова. – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 317 с. Режим доступу: http://repository.hneu.edu.ua/handle/123456789/23844.
4. Silberschatz Abraham, Galvin Peter B., Gagne Greg. Operating system concepts. 10th edition. – Wiley. – 2018. – 1278 p.
5. Sri Manikanta Palakollu. Practical System Programming with C: Pragmatic Example Applications in Linux and Unix-Based Operating Systems. – Apress. – 2020. – 286 p.
6. Pavel Yosifovich. Windows 10 System Programming, Part 1 and 2. – Leanpub. – 2021. – 1286 p.
7. Rodolfo Giometti. Linux Device Driver Development Cookbook. – Packt. – 2019. – 344 p.
8. Kerrisk Michael. The Linux Programming Interface. – No Starch Press, – 2010. – 1548 p.
9. Holcombe Jane, Holcombe Charles, Survey of operating systems. Sixth edition. – McGraw-Hill Education, – 2020. – 848 p.
10. Thomas Anderson, Michael Dahlin. Operating Systems: Principles and Practice. Vol.1-4. – Recursive Books. – 2015. – 600 p.
11. Bob Quinn, David Shute. Windows sockets network programming. – Addison-Wesley Professional. – 2010. – 328 p.
12. Операційні системи: Методичні вказівки до комп’ютерного практикуму. [Електронний ресурс] : навч. посіб. для студ. спец. 113 «Прикладна математика», 125 «Кібербезпека» / КПІ ім. Ігоря Сікорського ; уклад.: М. В. Грайворонський, В. В. Демчинський – Електронні текстові дані (1 файл: 1,44 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 74 с.

Методичні рекомендації до виконання лабораторних робіт

МОГИЛЬНИЙ Геннадій Анатолійович
СМАГІНА Ольга Олександрівна
ПЕРЕЯСЛАВСЬКА Світлана Олександрівна

ОПЕРАЦІЙНІ СИСТЕМИ ТА СИСТЕМНЕ ПРОГРАМУВАННЯ (ЧАСТИНА 1)

Методичні рекомендації до виконання лабораторних робіт
з освітнього компонента
для здобувачів освіти спеціальності
122– „Комп'ютерні науки"