Дисципліна «Операційні системи»

Лабораторна робота №14

Тема: «Основи керування віртуальними ОС»

Викладач: Олександр А. Блажко,

доцент кафедри ІС Одеської політехніки, blazhko@ieee.org

Мета роботи: придбання знань щодо основ використання апаратної віртуалізації та умінь із використання апаратної віртуалізації на прикладі програмного забезпечення *Oracle VM VirtualBox*.

1 Теоретичні відомості

1.1 Перша віртуальна ОС

Відомо, що *метою операційної системи* – є скорочення:

1) трудомісткості (складності) спілкування людини з комп'ютером або складність використання апаратних компонент комп'ютера;

2) часу простоювання апаратних компонент комп'ютера.

Віртуальна машина – модель обчислювальної машини як результат виконання методу абстрагування реальної обчислювальної машини, яка створена через абстрагування (віртуалізацію) обчислювальних ресурсів:

– процесора,

– оперативної пам'яті,

– пристроїв зберігання;

– вводу/виводу даних;

– комунікаційних каналів зв’язку.

Віртуальні машини поділяються на 2 головні категорії, в залежності від їх використання та відповідності до реальної апаратури:

– системні (апаратні) віртуальні машини, що забезпечують повноцінну емуляцію всієї апаратної платформи і підтримують виконання *віртуальної операційної системи*;

– прикладні віртуальні машини, які розроблені для виконання лише застосунків (прикладних програм), наприклад, віртуальна машина *Java*.

Системні віртуальні машини дозволяють розподіл апаратних ресурсів фізичної машини між різними копіями віртуальних машин, на кожній з яких може бути встановлена своя операційна система (ОС).

Вперше концепцію віртуальної ОС було реалізовано у системі *VM* (англ. *Virtual Machine*) для комп’ютера *IBM System/370* у 1972 році.

ЇЇ історія створення була пов`язана з тим, що у 1965 році, після випуску мейнфрейму *IBM System /360-67* ще не було багато-користувальницької ОС, яка б працювала у режимі з розподілом часу.

Компанія *IBM* традиційно знайша економне рішення проблеми:

– придбала авторські права на використання вже існуючої одно-користувальницької ОС *CMS* (*Cambridge Monitor System*);

– створила *CP/CMS* (*CP - Control Program*) – ОС для керування декількома копіями ОС *CMS* на одному комп'ютері *IBM System/360-67*;

– модифікувала *CP/CMS* до рівня *VM/CMS* (*Virtual Machine / Cambridge Monitor System*), яка здатна керуванти «віртуальними машинами» на комп'ютері *IBM System/370*.

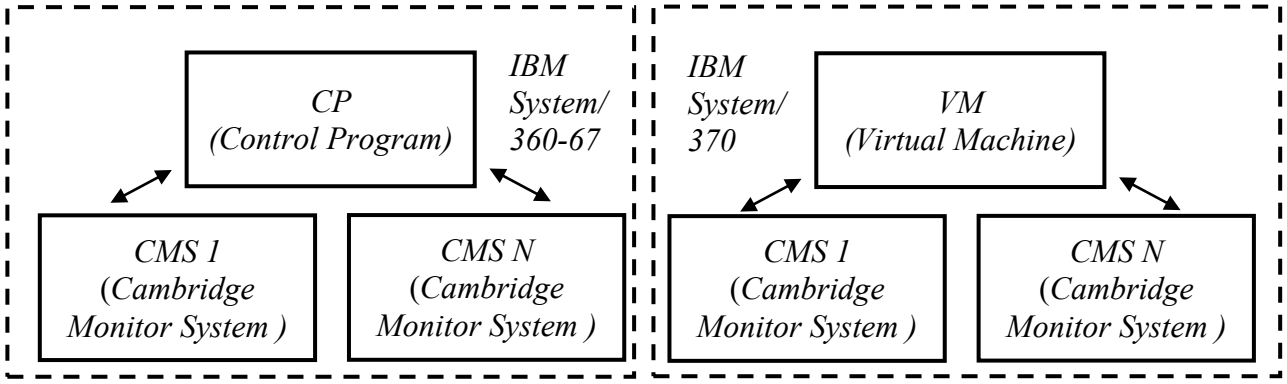


Рис. 1 – Робота *CP/CMS* та *VM/CMS*

VM-система складалась з:

– монітора віртуальних машин;

– зберігаємих ОС.

Монітор віртуальних машин надавав можливості завантажувати на модельованих віртуальних машинах інші ОС та забезпечував віртуалізацію та моделювання ресурсів.

Зберігаємі системи завантажувались із так званого *VM-образу* (*VM-image*), але не могли бути завантажені окремо від *VM*.

VM-система не була повноцінною ОС, а лише підтримувала роботу середовища для запуску інших ОС.

Сучасний приклад *VM*-систем – *гіпервізор,* програмне забезпечення, що виконує віртуалізацію.

При використанні гіпервізору ОС поділяються на 2 типи:

– *host*-ОС (англ. *host* – господарь) – ОС, яка встановлена на комп'ютері;

– *guest*-ОС (англ. *guest* – гість) – ОС, яка виконується віртуально на *host*-ОС.

Переваги використання віртуальної ОС:

– різні ОС можуть співіснувати на одному комп'ютері, і при цьому знаходитися в строгій ізоляції одна від одної;

– забезпечувати розширений набір машинних інструкцій, адже при моделюванні абстрактної обчислювальної машини набір інструкцій процесора віртуальної машини може бути довільним;

– широкі можливості контролю за програмами;

– легкість модифікацій та відновлення.

1.2 Програмне забезпечення апаратної віртуалізації *Oracle VM VirtualBox*

Розповсюдженим програмним забезпечнням апаратної віртуалізації є:

– *Oracle VM VirtualBox* – https://www.virtualbox.org/

– *VMware Workstation:*

- *Pro* – https://www.vmware.com/products/workstation-pro.html
- *Player* – https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html

1.2.1 Особливості апаратної віртуалізації *Oracle VM VirtualBox*

Особливості апаратної віртуалізації *Oracle VM VirtualBox*:

1) орієнтованість на різноманітність (гетерогенність) типів *host*-ОС та *guest*-ОС, що надає можливість створювати віртуальні ОС без обмежень;

2) гіпервізор постійно контролює всі операції *guest*-ОС, щоб вони не порушили роботу *host*-ОС, та перериває такі операції за необхідністю;

3) сучасні процесори *Intel* та *AMD* мають підтримку апаратної віртуалізації, яка суттєво підвищує швидкість роботи гіпервізору, – технології *Intel VT-x* та *AMD-V*, відповідно;

4) в системах апаратна віртуалізація може бути вимкнута на рівні *BIOS*, тоді під час встановлення віртуальної машини можливе таке повідомлення:

Not in a hypervisor partition (HVP=0) (VERR_NEM_NOT_AVAILABLE).

VT-x is disabled in the BIOS for all CPU modes (VERR_VMX_MSR_ALL_VMX_DISABLED).

у цьому випадку необхідно:

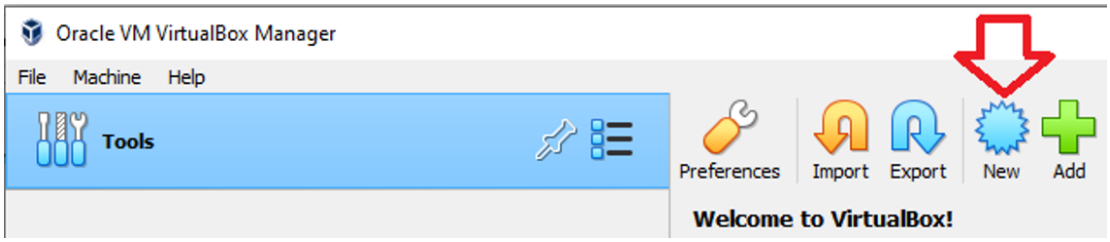
- під час включення комп’ютера зайти у *BIOS* та увімкнути підтримку апаратної віртуалізації;
- для ОС Windows перевірити включення опції *Hyper-V Platform*
- подробиці - https://support.lenovo.com/mn/uk/solutions/ht500006

5) для процесорів без технологій *Intel VT-x* та *AMD-V* можлива програмна емуляція апаратної віртуалізації.

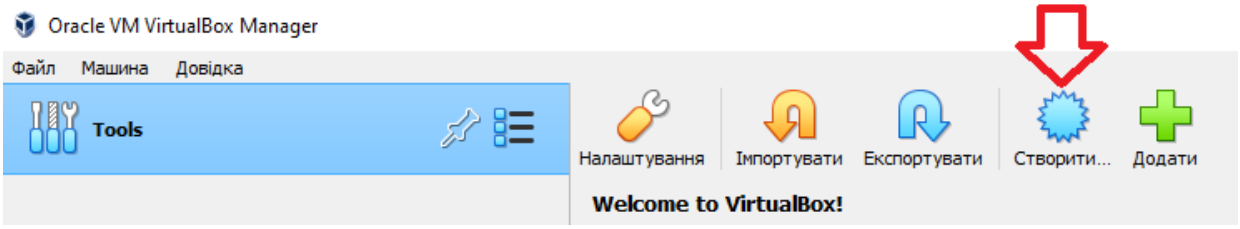
1.2.2 Завантаження програмного забезпечення *Oracle VM VirtualBox*

Посилання на отримання останньої версії *Oracle VirtualBox*, наприклад, версії 7 – https://www.virtualbox.org/wiki/Downloads

Після встановлення програми можна змінити мову інтерфейсу, наприклад на українську, перейшовши в розділ «*Preference*»/«*Language*». Далі можна почати налаштування віртуальних машин, як показано на рисунку 2, через перехід за кнопкою *«New»/«Створити»*.



(a) англомовний інтерфейс головного екрану програми *VirtualBox*



(b) україномовний інтерфейс головного екрану програми *VirtualBox*

Рис. 2 – Приклад головного екрану програми *VirtualBox*

1.2.3 Завантаження *ISO*-образу з інсталяцією ОС

Для створення віртуальної машини програма *VirtualBox* найчастіше потребує файл з інсталяцією ОС у вигляді *ISO-образу*.

ISO-образ (*ISO-image*) – це файл, який використовується для запису на *CD/DVD*-диски для розповсюдження інсталяційних пакетів програм, але сьогодні він найчастіше використовується за межами *CD/DVD*.

Для проведення навчальних експериментів рекомендується використати старі пакети встановлення *ОС Linux* невеликого розміру, які не використовують графічний режим роботи.

Посилання на отримання *ISO*-образів *Server Linux Ubuntu* 14.04.6 релізу (версії), який має розмір 623 Мб, можна отримати на сторінці http://www.releases.ubuntu.com/14.04/

Традиційно у назві *ISO*-образу для 32-бітного варіанту ОС міститься фраза *i386*, як сімейство процесорів *Intel*, а для 64-бітного варіанту ОС міститься фраза *amd64*:

- 32-бітна ОС – http://www.releases.ubuntu.com/14.04/ubuntu-14.04.6-server-i386.iso
- 64-бітна ОС – http://www.releases.ubuntu.com/14.04/ubuntu-14.04.6-server-amd64.iso

1.2.4 Керування віртуальними машинами через графічний інтерфейс

Після натискання кнопки «Створити» створюється опис віртуальної машини, як показано на рисунку 3, де вказано наступне:

– назва віртуальної машини – *«Ubuntu for Blazhko»*;

– системний каталог з файлами віртуальної машини (можна залишити як вказано);

– посилання на *ISO*-образ – *C:\Users\blazhko\ubuntu-14.04.6-server-i386.iso*

– прапорець *«Skip unattended installation»*.

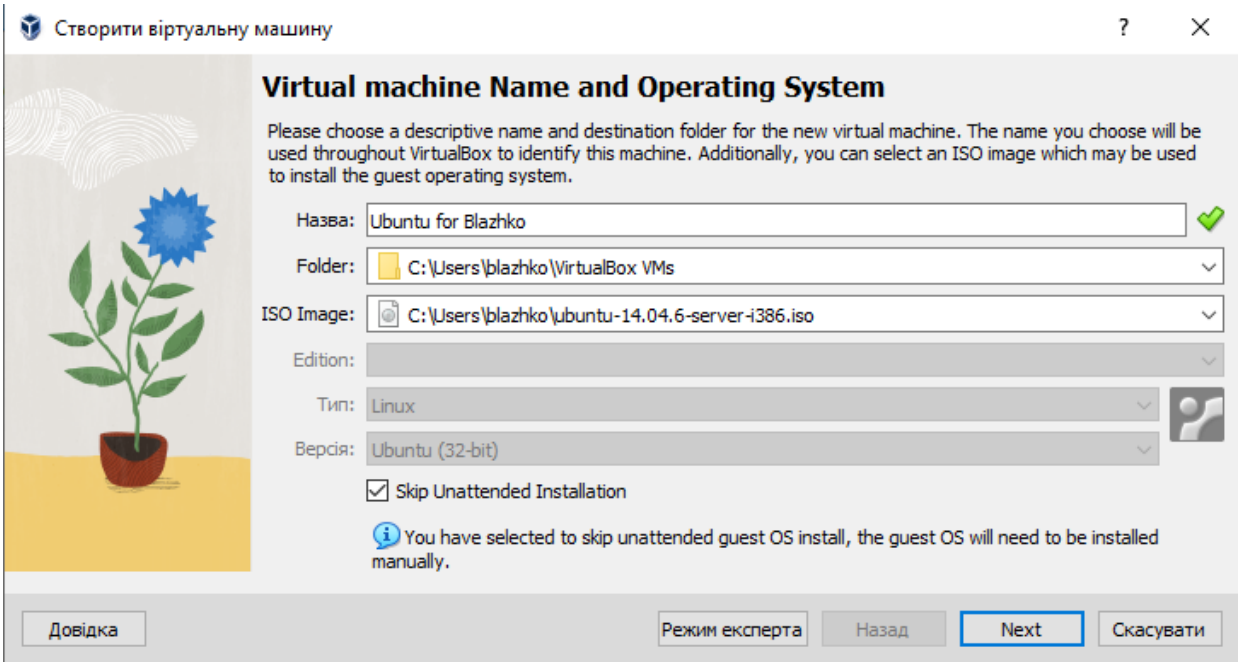


Рис. 3 – Приклад екранної форми детальної настройки ОС в розділі «Екран»

На рисунку 4 наведено приклад екранної форми підключення пам’яті та процесору.

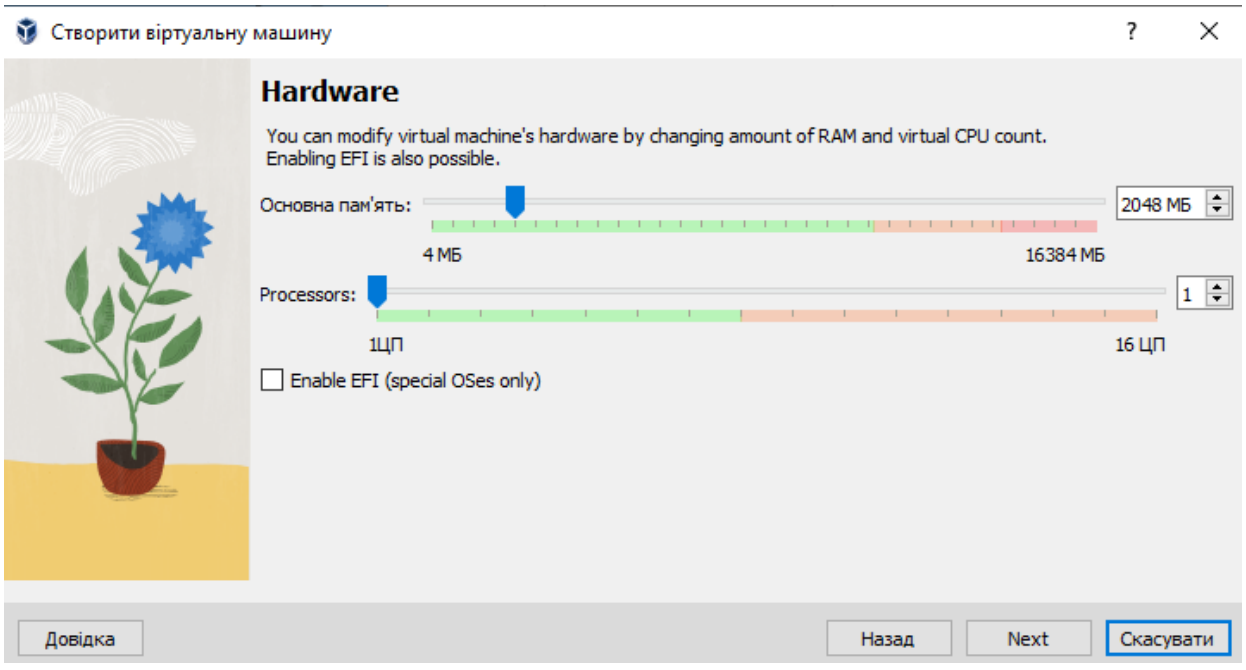


Рис. 4 – Приклад екранної форми підключення пам’яті та процесору

На рисунку 5 наведено приклад екранної форми підключення віртуального диску

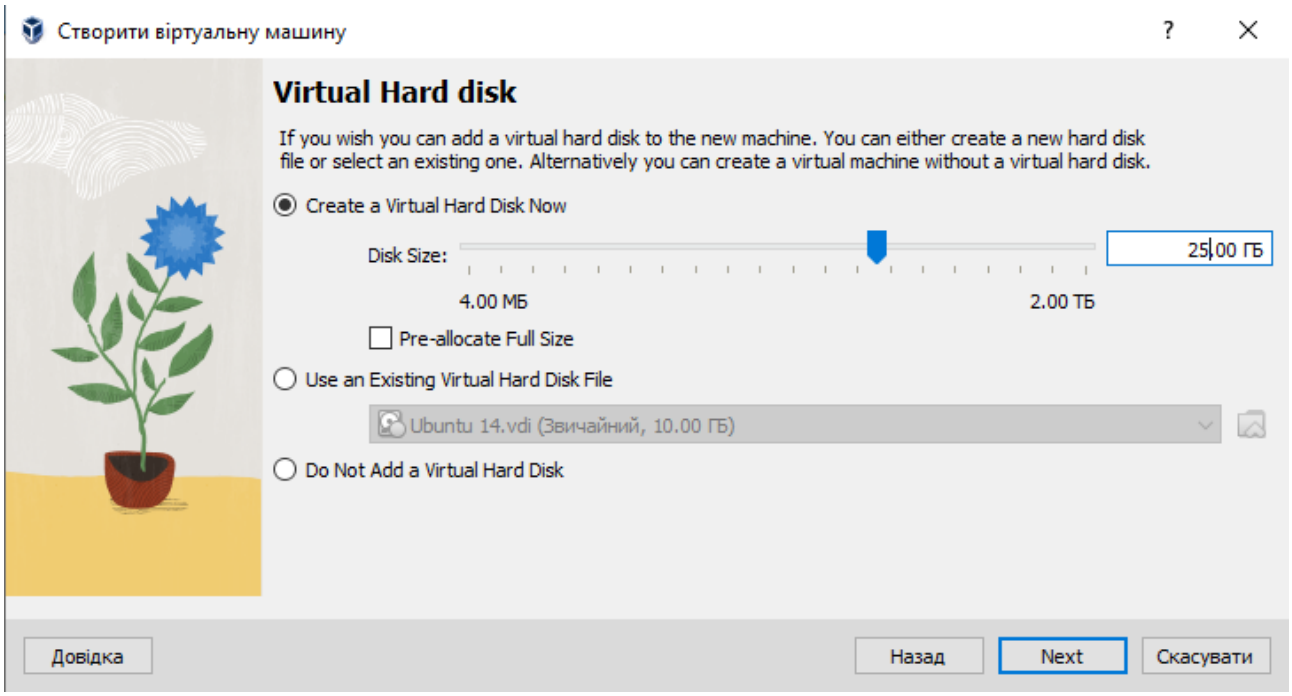


Рис. 5 – Приклад екранної форми підключення віртуального диску

На рисунку 6 наведено приклад екранної форми опису створенної віртуальної машини.

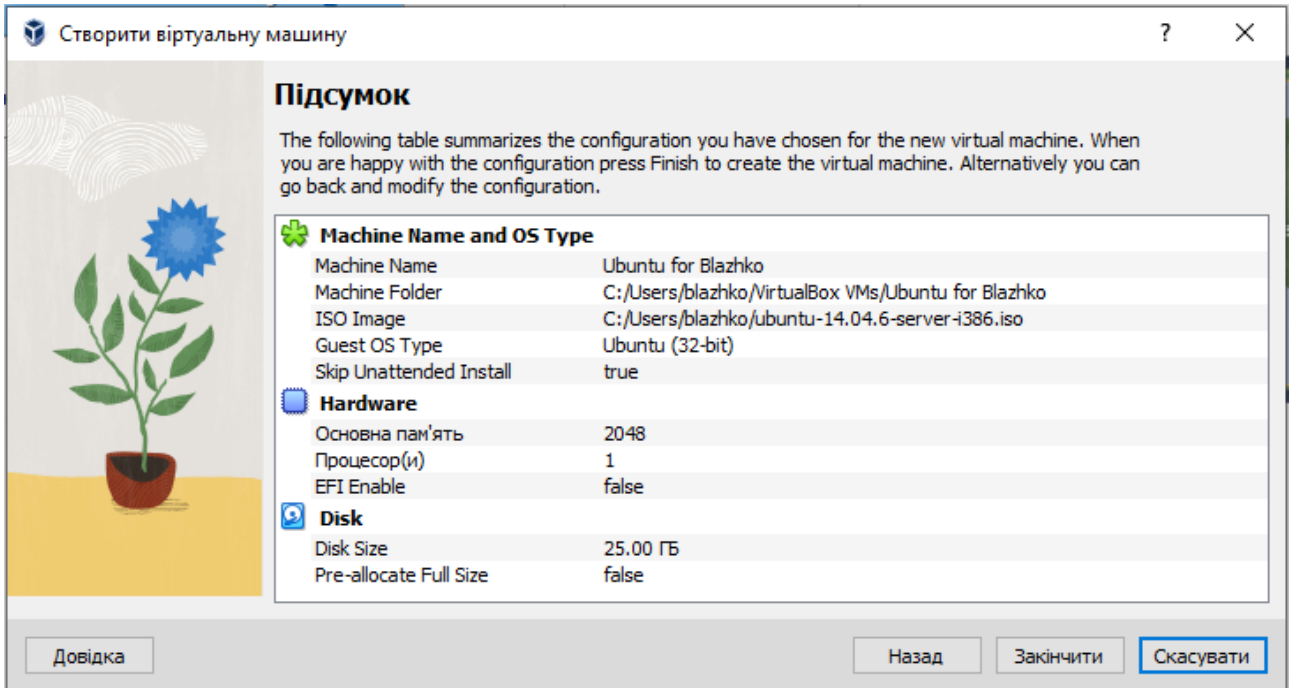


Рис. 6 – Приклад екранної форми опису створенної віртуальної машини.

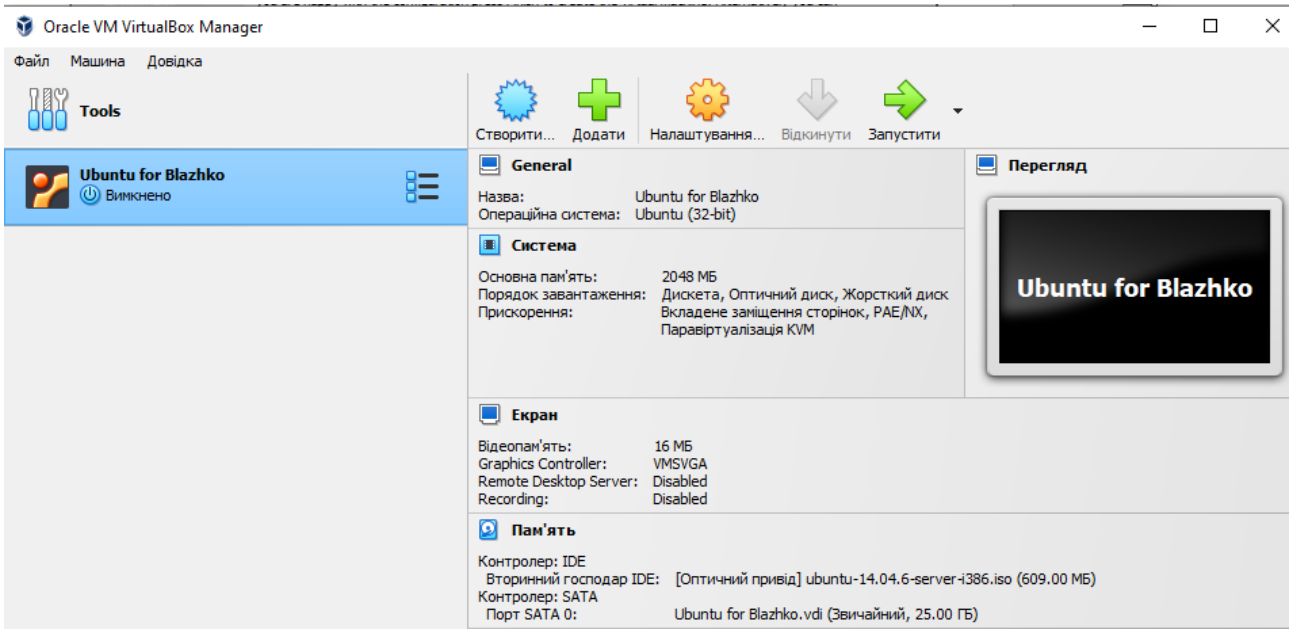


Рис. 7 – Приклад екранної форми

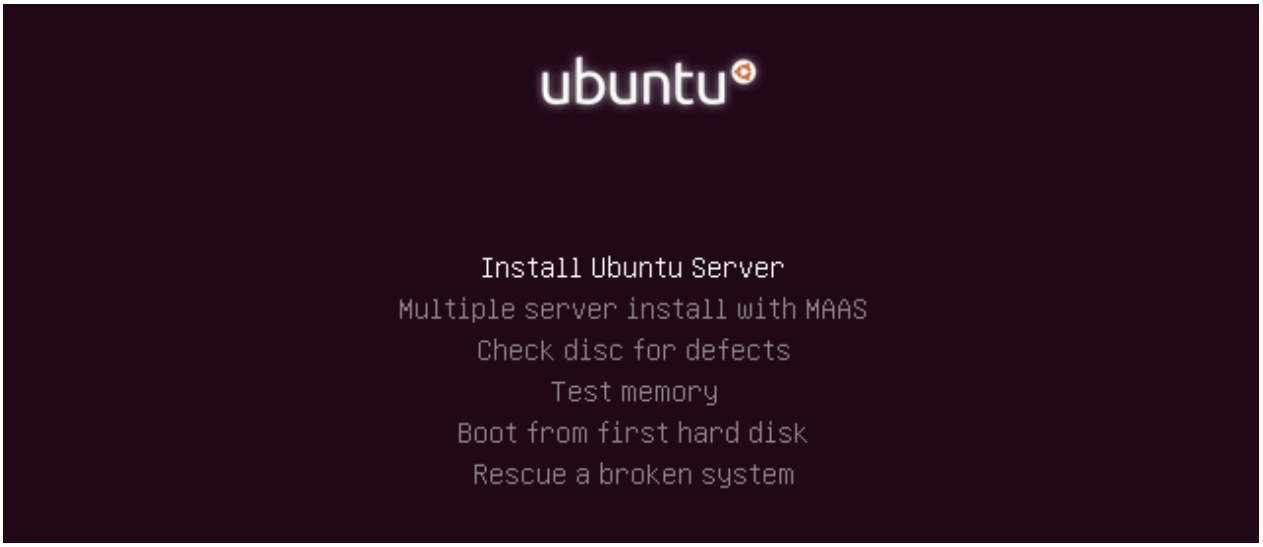


Рис. 8 – Приклад екранної форми

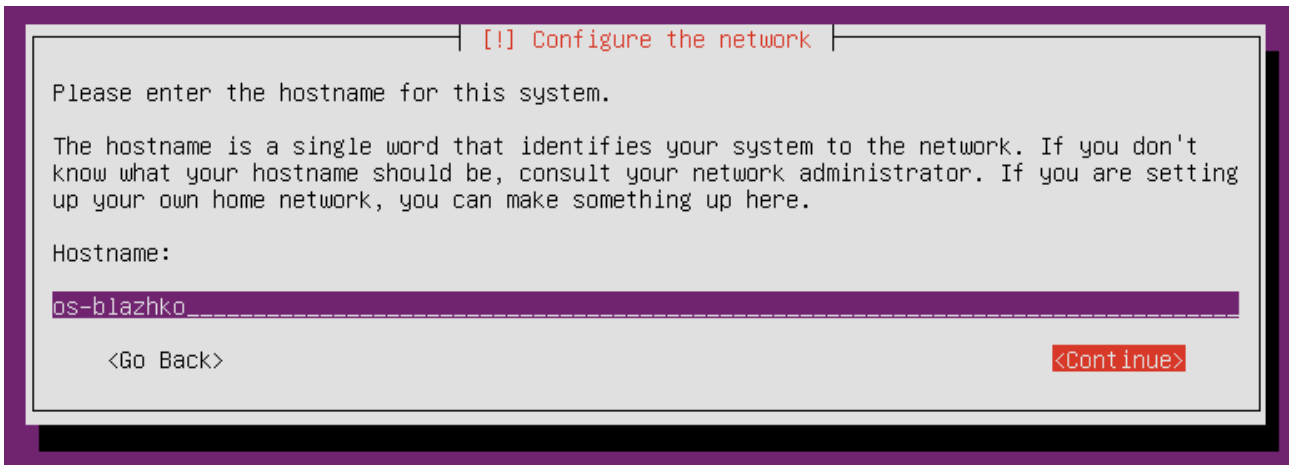


Рис. 9 – Приклад екранної форми

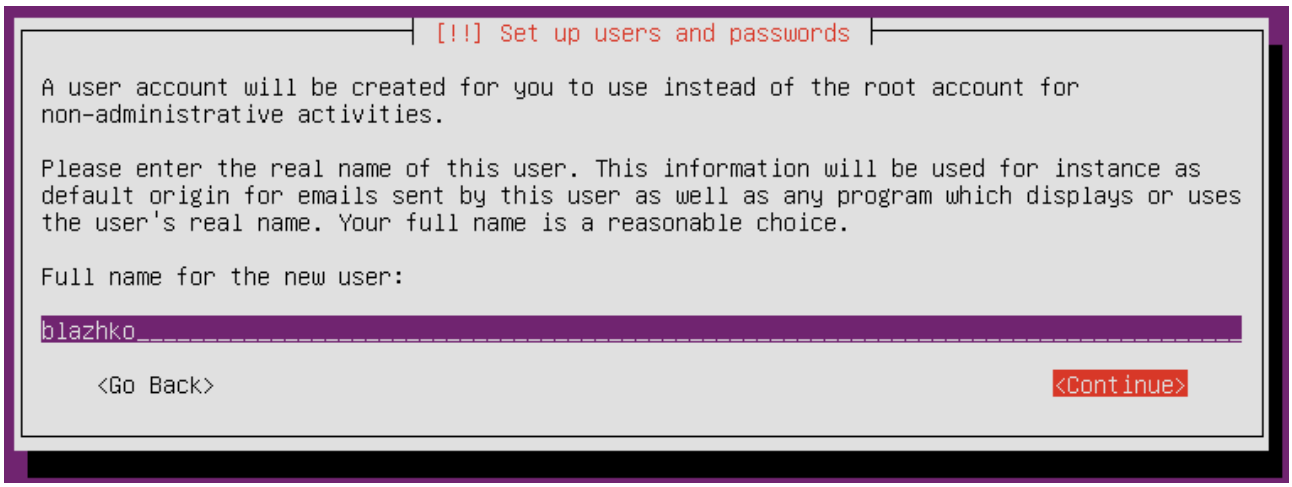


Рис. 10 – Приклад екранної форми

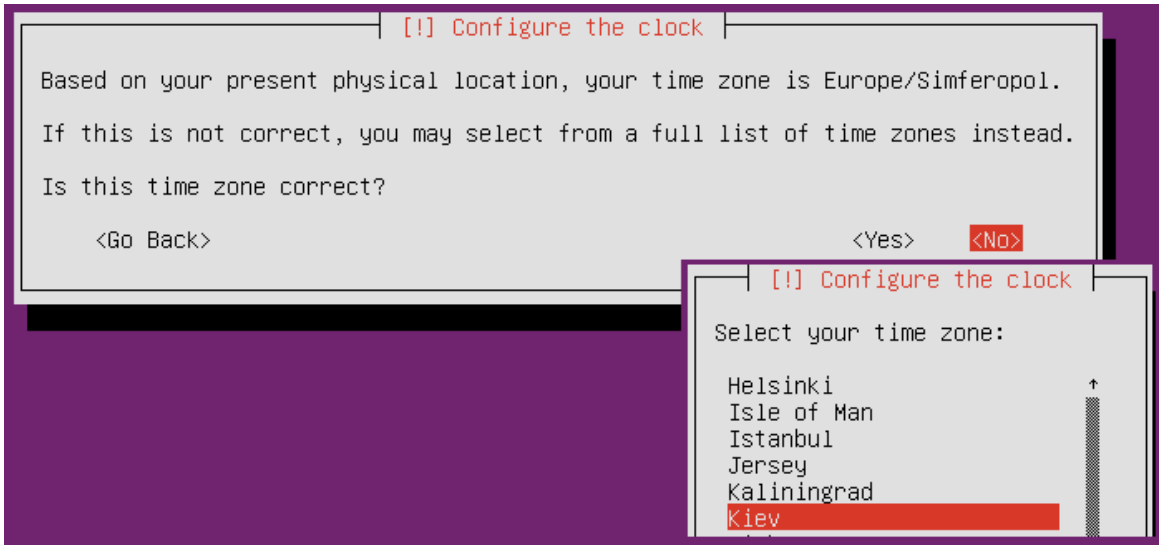


Рис. 11 – Приклад екранної форми

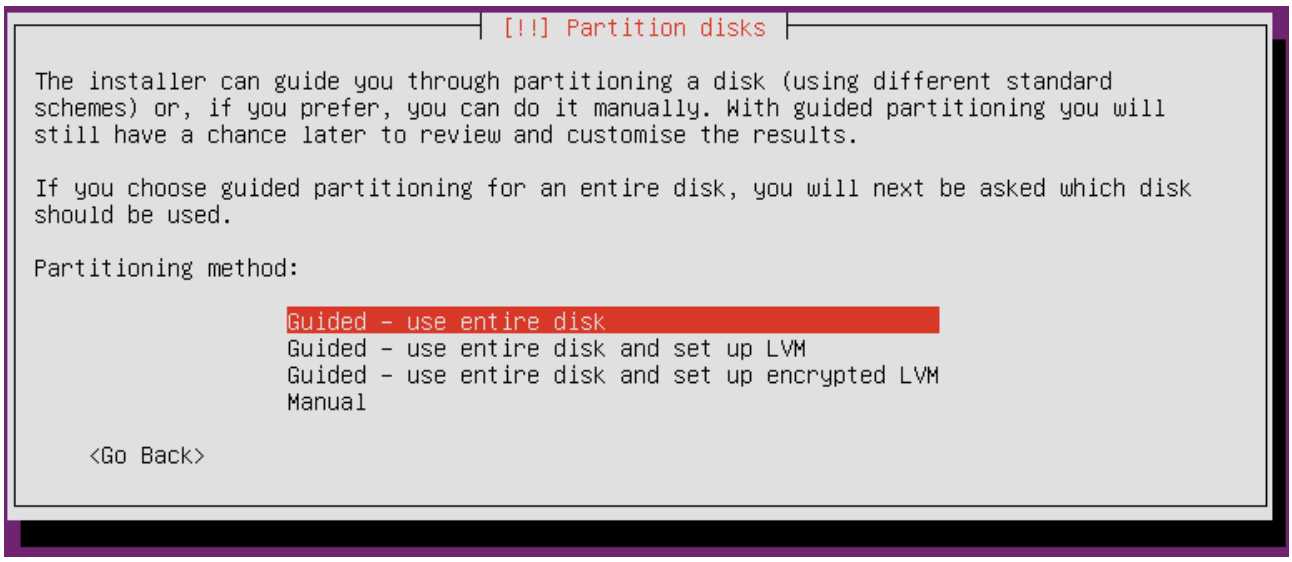


Рис. 12 – Приклад екранної форми

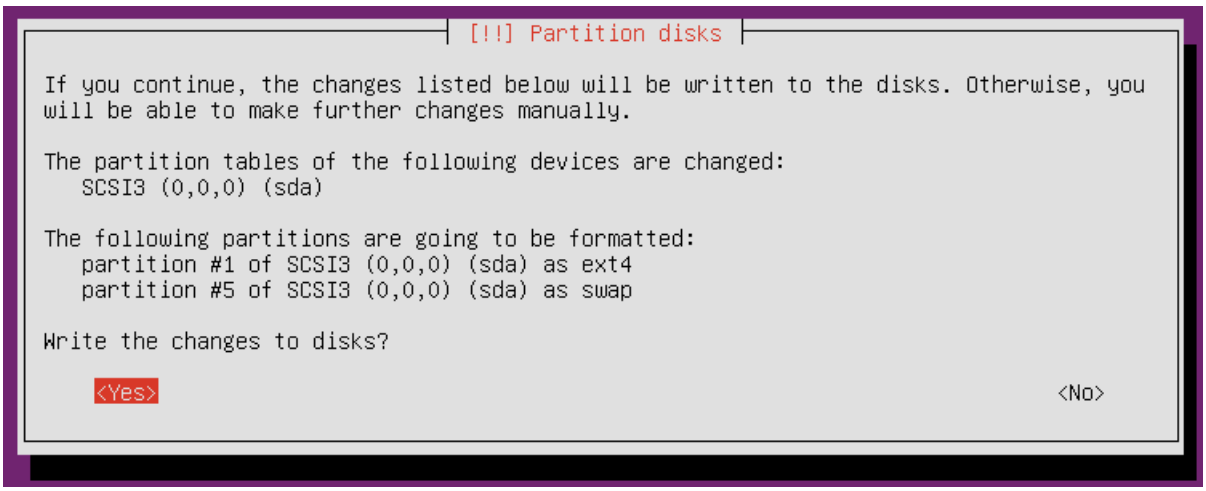


Рис. 13 – Приклад екранної форми

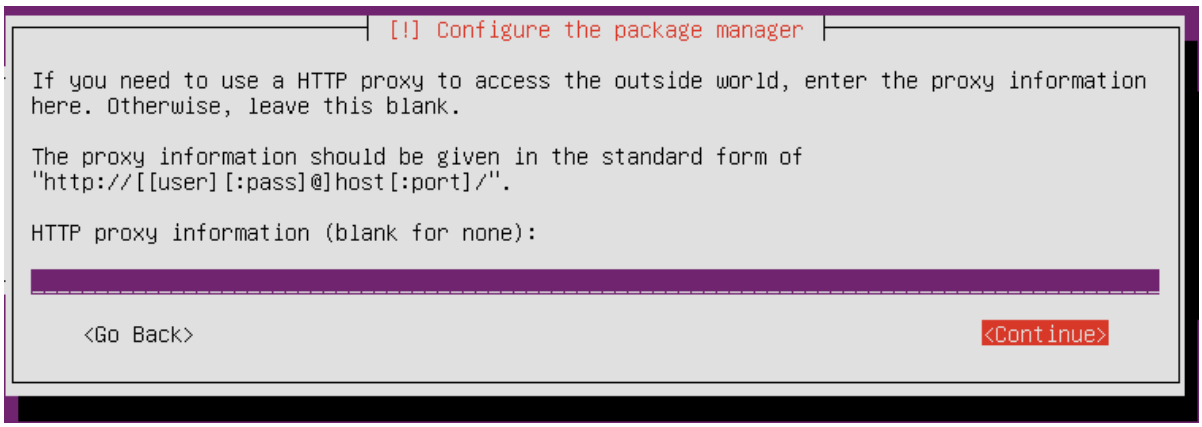


Рис. 14 – Приклад екранної форми

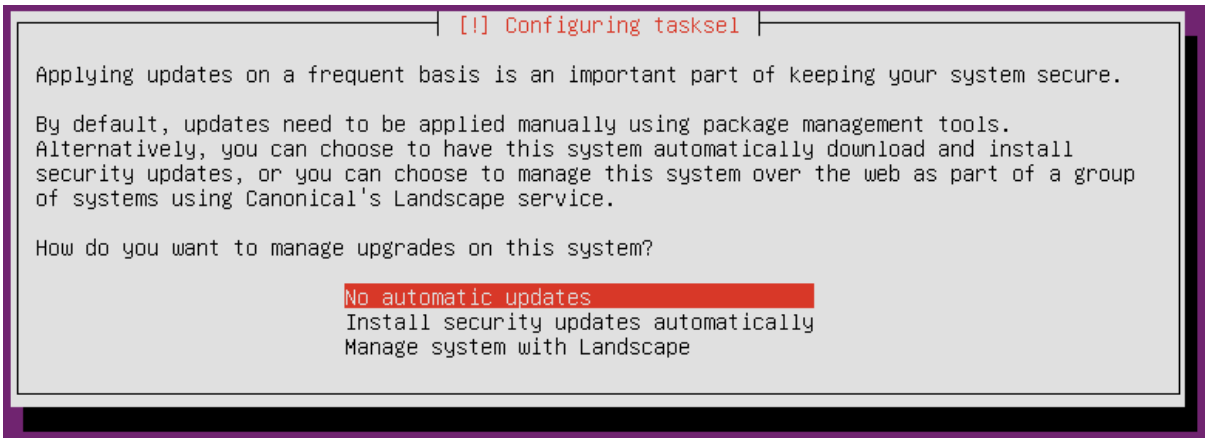


Рис. 15 – Приклад екранної форми

На рисунку 16 наведено фрагмент екранної форми із визначенням програми «*OpenSSH Server*», яка необхідна ля встановлення в ОС процесу-даймону підтримки *SSH*-протоколу.

Рис. 16 – Приклад екранної форми

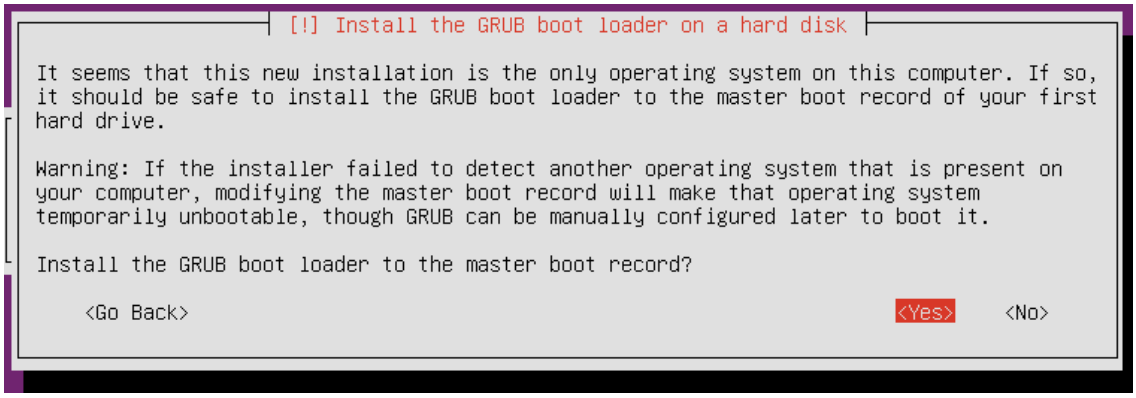


Рис. 17 – Приклад екранної форми

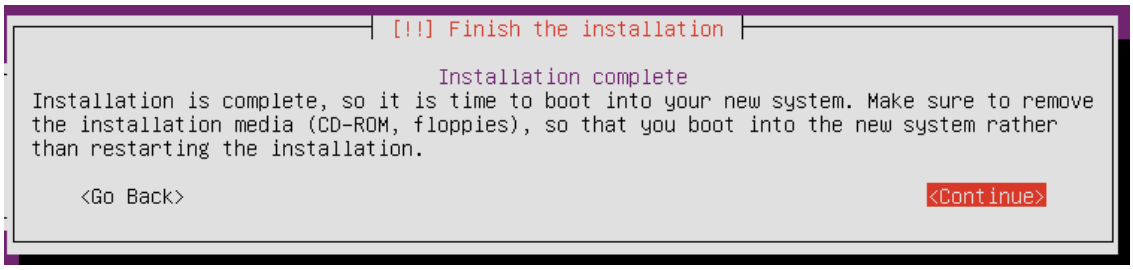


Рис. 18 – Приклад екранної форми

Після перезавантаження віртуальної машини виконується старт віртуальної машини та завантаження віртуальної ОС. На рисунку 19 наведено приклад екранної форми із псевдотерміналом доступу до ОС. Програма дозволяє перемикати роботу користувача серед шести псевдотерміналів, використовуючи комбінацію клавіш *Alt-F1, Alt-F2, Alt-F3, Alt-F4, Alt-F5, Alt-F6*.

За комбінацією клавіш *Alt-F7* можна перейти у псевдотермінал із повдомленнями під час запуску віртуальної машини.

Рис. 19 – Приклад екранної форми із псевдотерміналом доступу до ОС

Робота у вказаних псевдотерміналах може мати недоліки стосовно операцій копіювання рядків між *host*-ОС та *guest*-ОС. Тому в подальшому рекомендується працювати з віртуальною машиною через зовнішній доступ через *Git-Bash*-псевдотермінали та *SSH*-протокол. Для встановлення з'єднання з *Git-Bash*-псевдотермінала через *SSH*-протокол до віртуальної машини необхідно додатково створити правило *Port Forwarding*, приклад якого представлено на рисунку 20.

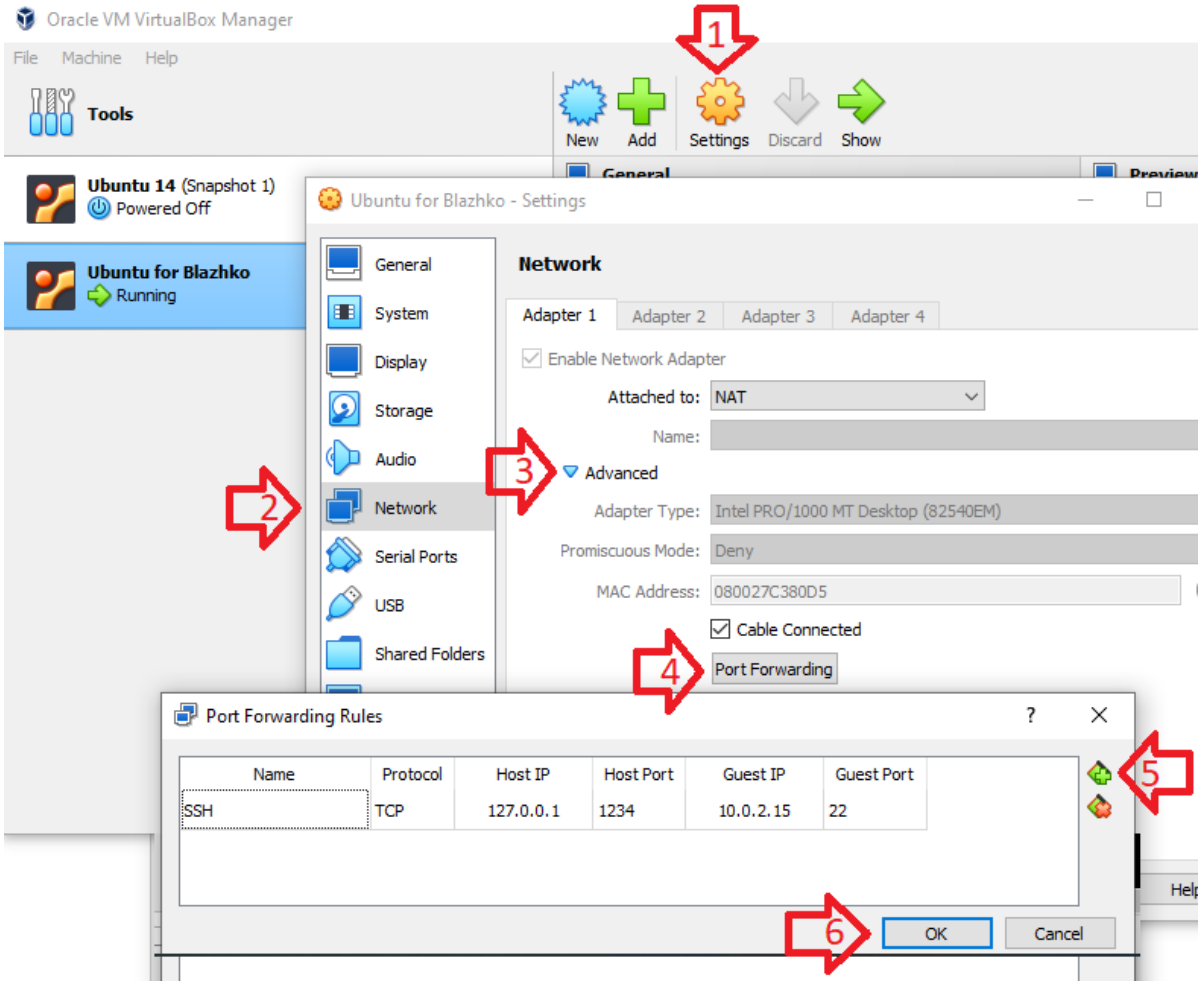


Рис. 20 – Приклад екранної форми

За замовчуванням *VirtualBox* встановлює для *IP*-адреси віртуальної машини (*Guest IP*) значення = 10.0.2.15. Але це значення можна переглянути, виконавши у віртуальній машині команду:

```
sudo ifconfig
```

Команда виведе на екран дані про мережеву карту *eth0*, серед яких буде параметр *inet addr* із необхідним значенням *Guest IP*.

Після налаштування правила мережевого перенаправлення можна встановити з'єднання із віртуальною машиною, виконавши команду:

```
ssh -p 1234 blazhko@127.0.0.1
```

Використання *SSH*-протоколу можливе, якщо ви не забули вказати прапорець для «*OpenSSH server»* під час встановлення ОС. Але якщо забули, тоді треба, знаходячись в командному рядку віртуальної машини, додати програмний пакунок через команду:

```
sudo apt install openssh-server
```

1.2.5 Керування віртуальними машинами через інтерфейс командного рядку

Для керування віртуальними машинами через інтерфейс командного рядку використовується утиліта *VboxManage*.

Нижче вказано основні кроки створення віртуальної машини.

Крок 1. Реєстрація віртульної машини

Після інсталяції програми *VirtualBox* утиліта *VboxManage* може бути доступною в командному рядку за замовчуванням, але якщо це не сталося, наприклад, в ОС *Windows*, у *Git-Bash*-оболонці треба додати такий шлях до *executable*-файлів до змінної *PATH*:

```
export PATH=$PATH:"/c/Program Files/Oracle/VirtualBox"
```

Для реєстрації віртуальної машини необхідно визначитися із типом ОС. Для отримання переліку наявних типів ОС або дистрибутивів ОС використовується команда:

```
VBoxManage list ostypes
```

Через значну кількість типів ОС можна виконати їх фільтрацію, наприклад, типу *Ubuntu*, через конвеєр командою *grep*:

```
VBoxManage list ostypes | grep "Ubuntu"
```

Для реєстрації віртуальної машини використовується команда:

VBoxManage createvm

Основні опції:

--*name* – назва віртуальної ОС;

--*ostype* – тип ОС або назва дистрибутиву ОС;

--*register* – реєстрація віртуальної машини.

Наприклад, для реєстрації віртуальної машини з назвою *"Ubuntu for Blazhko"* дистрибутиву *Ubuntu14_LTS* використовується команда:

```
VBoxManage createvm --name "Ubuntu for Blazhko" \
-ostype Ubuntu14_LTS --register
```

Після реєстрації віртуальній машині надається унікальний ідентифікатор *UUID,* за яким можна керувати, використовуючи його замість назви, що ефективно для автоматизації процесів розгортання десятків віртуальних машин.

Для отримання списку зареєстрованих віртуальних машин використовується команда:

```
VBoxManage list vms
```

Для отримання списку запущених віртуальних машин використовується команда:

```
VBoxManage list runningvms
```

Крок 2. Встановлення загальних параметрів віртуальної машини

Віртуальній машині як будь-якому комп'ютеру необхідні віртуальні ресурси:

- процесор;
- оперативна пам'ять;
- графічна карта;
- мережева карта;
- зовнішні пристрої.

Для встановлення параметрів віртуальнї машини використовується команда:

```
VBoxManage modifyvm
```

Приклади параметрів, які найчастіше використовуються:

--cpus count – кількість ядер процесору;

--memory size – розмір оперативної пам'яті (в мегабайтах);

--vram size – розмір оперативної пам`яті графічної карти;

--nicN type – мережева карта з номером *N* та типом *type.*

Наприклад, для встановлення у віртуальній машині *"Ubuntu for Blazhko"* двоядерного процесору необхідно виконати команду:

```
VBoxManage modifyvm "Ubuntu for Blazhko" --cpus 2
```

Наприклад, для встановлення у віртуальній машині *"Ubuntu for Blazhko"* розміру оперативної пам'яті = 2048 Mb необхідно виконати команду:

```
VBoxManage modifyvm "Ubuntu for Blazhko" --memory 2048
```

Наприклад, для встановлення у віртуальній машині *"Ubuntu for Blazhko"* розміру графічної пам'яті = 128 Mb необхідно виконати команду:

```
VBoxManage modifyvm "Ubuntu for Blazhko" --vram 128
```

Наприклад, для підключення до віртуальної машини першої мережевої карти з конфігурацією *NAT (Network Address Translation)*:

```
vboxmanage modifyvm "Ubuntu for Blazhko" --nic1 nat
```

Наприклад, для створення правила *natph1* (*ph – Port Forwarding*) на першу мережеву карту *nat1*, яке буде перенаправляти мережеві пакети для з'єднання через *SSH*-протокол із

guest-ОС (*IP-address=10.0.2.15, SSH-port=22*) на *IP-адресу=127.0.0.1* з використанням *SSH-port=1234*, необхідно виконати команду:

```
VBoxManage modifyvm "Ubuntu for Blazhko" \
--natpf1 "SSH,tcp,127.0.0.1,1234,10.0.2.15,22"
```

Одними з прикладів інтерфейсів зовнішніх пристроїв є:

– *ATA* (*Advanced Technology Attachment*) або *IDE* (*Integrated Drive Electronics*) – паралельний інтерфейс обміну даними з накопичувачами інформації, наприклад, *Hard*-дисками та *CD/DVD-ROM*-приводами;

– *SATA* (*Serial ATA*) – послідовний інтерфейс обміну даними з накопичувачами.

Для підключення контролера інтерфейсу зовнішніх пристроїв використовується команда:

```
VBoxManage storagectl <VM-name>
```

--name controller-name – довільна назва контролеру;

--add=system-bus-type – інтерфейс підключення зовнішніх пристроїв, наприклад, *ide, sata, scsi, usb*;

--controller=chipset-type – контролер інтерфейс підключення зовнішніх пристроїв, наприклад, *PIIX4 (PCI IDE ISA Xcelerator)*, *IntelAHCI (Advanced Host Controller Interface for SATA)*.

Наприклад, для підключення до віртуальної машини *"Ubuntu for Blazhko"* контролеру *IntelAHCI* з інтерфейсом *sata*-типу за назвою *MySATA*, необхідно виконати команду:

```
VBoxManage storagectl "Ubuntu for Blazhko" --name "mySATA" \
--add sata --controller IntelAHCI
```

Наприклад, для підключення до віртуальної машини *"Ubuntu for Blazhko"* контролеру *PIIX4* з інтерфейсом *ide*-типу за назвою *"myIDE "*:

```
VBoxManage storagectl "Ubuntu for Blazhko" --name "myIDE" \
--add ide --controller PIIX4
```

Для створення віртуального сховища даних використовується команда:

```
VBoxManage createmedium
```

Основні опції:

--filename – шлях розташування файлу;

--size – розмір файлу (Мбайт);

--format – тип файлу, наприклад, VDI.

Наприклад, для створення віртуального сховища даних розміром *10240 Mb*, яке буде розміщено у файлі *disk01.vdi*, необхідно виконати команду:

```
VBoxManage createmedium --filename disk01.vdi --size 10240
```

Для підключення віртуального сховища до контролеру інтерфейсу із зовнішніми пристроями використовується команда:

```
VBoxManage storageattach <VM-name>
```

Опції команди:

--storagectl name – назва контролеру;

--type=dvddrive | fdd | hdd – тип контролеру

--medium MediumPath – шлях до файлу з віртуальним сховищем;

--remove – відключити раніше підключене віртуальен сховище.

Наприклад, для підключення у віртуальній машині *"Ubuntu for Blazhko"* до контролеру з назвою "*mySATA*" віртуального сховища *disk01.vdi hdd*-типу, на якому в подальшому буде розташувано файлову систему віртуальної машини, необхідно виконати команду:

```
VBoxManage storageattach "Ubuntu for Blazhko" \
--storagectl "mySATA" --port 0 --device 0 --type hdd \
--medium disk01.vdi
```

Наприклад, для підключення у віртуальній машині *"Ubuntu for Blazhko"* до контролеру з назвою *"myIDE"* файлу з *ISO*-файлу *dvddrive*-типу *ubuntu-14.04.6-server-i386.iso* з інсталяцією ОС *Linux* необхідно виконати команду:

```
VBoxManage storageattach "Ubuntu for Blazhko" \
--storagectl "myIDE" --port 0 --device 0 --type dvddrive \
--medium ubuntu-14.04.6-server-i386.iso
```

Крок 3. Запуск віртуальної машини

Для отримання опису віртуальної машини використовується команда:

```
VBoxManage showvminfo VM-name
```

Наприклад, для отримання опису віртуальної машини *"Ubuntu for Blazhko"* необхідно виконати команду:

```
VBoxManage showvminfo "Ubuntu for Blazhko"
```

Для запуску віртуальної машини використовується команда:

```
VBoxManage startvm VM-name [--type headless]
```

Опція *--type headless* дозволяє запустити віртуальну машину без графічного режиму.

Наприклад, для запуску віртуальної машини *"Ubuntu for Blazhko"* необхідно виконати команду:

```
VBoxManage startvm "Ubuntu for Blazhko"
```

Для виключення віртуальної машини використовується команда:

```
VBoxManage controlvm VM-name poweroff
```

Наприклад, для виключення віртуальної машини *"Ubuntu for Blazhko"* необхідно виконати команду:

```
VBoxManage controlvm "Ubuntu for Blazhko" poweroff
```

Для призупинки роботи віртуальної машини зі збереженням поточного стану роботи використовується команда:

```
VBoxManage controlvm <vm-name> savestate
```

Наприклад, для призупинки роботи віртуальної машини *"Ubuntu for Blazhko"* зі збереженням поточного стану роботи необхідно виконати команду:

```
VBoxManage controlvm "Ubuntu for Blazhko" savestate
```

Крок 4. Зняття з реєстрації віртуальної машини

Для зняття з реєстрації віртуальної машини та видалення всіх пов'язаних з нею файлів використовується команда:

```
VBoxManage unregistervm <vm-name> [--delete-all]
```

Наприклад, для зняття з реєстрації віртуальної машини *"Ubuntu for Blazhko"* необхідно виконати команду:

```
VBoxManage unregistervm "Ubuntu for Blazhko" --delete-all
```

1.3 Адміністративне керування в ОС Linux

Всі команди адміністрування може виконувати користувач з правами адміністратора.

Для ОС *Ubuntu*, наприклад, рівень адміністратора можна визначити, вказавши попередньо команду *sudo*.

1.3.1 Команди керування групами користувачів

Команда створення групи *groupadd*:

```
groupadd ім`я_групи прапорці_команди
```

Наприклад, для створення групи AI221, необхідно виконати команду:

```
groupadd AI221
```

Створюється обліковий запис групи у файлі */etc/group* з наступною структурою:

ім`я_групи : x : унікальний_ідентифікатор_групи : імена_користувачів_через_кому

Для перегляду вмісту файлу можна скористатися командою *less*:

```
less /etc/group
```

Подробиці про використання прапорців команди можна отримати через команду:

```
man groupadd.
```

Для внесення змін в опис групи використовується команда *groupmod*.

Для видалення групи використовується команда *groupdel*:

```
groupdel ім`я_групи.
```

1.3.2 Команди керування користувачами

Команда створення користувачів:

useradd ім`я_користувача опції

Основні опції:

– *c* – реальне ім'я користувача;

– *d* – каталог користувача;

– *m* – створення каталогу користувача, якщо він відсутній, та копіювання в нього файлів конфігурації з каталогу */etc/skel*

– *g* – група користувача;

– *u* – ідентифікатор користувача;

– *s* – шлях до оболонки командного рядка, наприклад, */bin/bash*

Для зручності адміністрування всі домашні каталоги користувачів створюються в каталозі */home*, а їхні імена збігаються з ім'ям користувача.

Наприклад, для реєстрації співробітника Шевченко Олександра з ім`ям користувача *shevchenko*, який входить до групи *AI221*, має домашній каталог */home/shevchenko* та використовує для підключення оболонки командного рядка програму */bin/bash,* використовується наступна команда:

```
useradd shevchenko -c "Шевченко Олександр" -s /bin/bash \
-d /home/shevchenko -g AI221 -m
```

Команда редагування параметрів користувача:

usermod ім`я_користувача прапорці

Прапорці команди збігаються з опціями команди *useradd*.

Для доступу до оболонки ОС під ім`ям нового користувача виконується команда:

su - ім`я_користувача.

Наприклад:

su - shevchenko

Ключ «–» вказує ОС читати конфігураційні файли, які зберігаються у каталозі користувача, насамперед файл конфігурації оболонки командного рядку.

Для виходу користувача із оболонки ОС використовується команда:

exit або *logout*

Команда видалення користувача:

userdel ім`я_користувача

1.3.3 Встановлення СКБД *PostgreSQL* доступної версії в ОС

Для встановлення, оновлення і вилучення програмних пакунків в ОС *Debian* та в ОС, які засновані на ОС *Debian*, наприклад, *Ubuntu, Linux Mint*, використовується програма *apt (Advanced Packaging Tool)*.

Приклади команд програми *apt*:

sudo apt update – оновлення баз даних пакунків (вказаних в */etc/apt/sources.list*)

sudo apt upgrade – оновлення ОС;

apt search пакунок – пошук пакунків;

apt search ^пакунок – пошук пакунків за регулярним виразом;

sudo apt install пакунок – встановити пакунок;

sudo apt remove пакунок – зняти з реєстрації пакунок.

sudo apt purge пакунок – видалити пакунок з усіма файлами

В *ОС Ubuntu 14* для швидкого встановлення з пакунків доступна СКБД *PostgreSQL* лише версії 9.3.

Наприклад, щоб встановити СКБД *PostgreSQL*, необхідно виконати команду:

```
sudo apt install postgresql
```

Після встановлення СКБД *PostgreSQL* можна увійти під обліковим записом *postgres*:

```
sudo su - postgres
```

Для створення нової БД СКБД *PostgreSQL* використовується команда:

```
created db-name
```

Для створення нового користувача СКБД *PostgreSQL* використовується команда:

```
createuser user-name
```

Для зупинки сервера СКБД *PostgreSQL* необхідно виконати команду:

```
/usr/lib/postgresql/9.3/bin/pg_ctl \
-D /var/lib/postgresql/9.3/main/ stop
```

Для видалення пакунку СКБД *PostgreSQL* необхідно виконати команду:

```
sudo apt purge postgresql
```

1.3.4 Встановлення останньої версії СКБД *PostgreSQL*

Для встановлення останньої версії СКБД *PostgreSQL*, наприклад, версії 15, необхідно виконати процес побудови всіх програм на основі сирцевих кодів (*source code*), використовуючи утиліту *make*. Нижче наведено основні кроки, які дозволяють встановити в *ОС Ubuntu* СКБД *PostgreSQL* версії 15.3.

Крок 1. Встановити пакунки з бібліотеками, від яких залежать програмні модулі СКБД *PostgreSQL*

```
sudo apt install gcc libreadline-dev zlib1g-dev make
```

Крок 2. Перейти до класичного каталогу зберігання сирцевих кодів

```
cd /usr/local/src
```

Крок 3. Отримати архів із сирцевими кодами СКБД *PostgreSQL* версії 15.3

```
sudo wget \
https://ftp.postgresql.org/pub/source/v15.3/postgresql-15.3.tar.gz
\ --no-check-certificate
```

Крок 4. Розкрити архів

```
sudo tar xvzf postgresql-15.3.tar.gz
```

Крок 5. Перейти до каталогу

```
cd postgresql-15.3
```

Крок 6. Виконати команду конфігурування *make*-файлу із вказуванням каталогу */usr/local/pgsql-15* майбутнього розташування скомпільованих програм СКБД *PostgreSQL*

```
sudo ./configure –prefix=/var/lib/postgresql/15.3
```

Крок 7. Побудувати програми СКБД *PostgreSQL:*

```
make
```

Крок 8. Встановити програми, які будуть розташовані в каталозі */usr/local/pgsql15*

```
sudo make install
```

Крок 9. Створити каталог для розміщення файлів бази даних

```
sudo mkdir /var/lib/postgresql/15.3/data
```

Крок 10. Визначити адміністратора СКБД власником каталогу:

```
sudo chown postgres.postgres /var/lib/postgresql/15.3/ –R
```

Крок 11. Перемкнути псевдотермінал на роботу із обліковим записом користувача *postgres* як адміністратора СКБД *PostgreSQL:*

```
sudo su – postgres
```

Крок 12. Встановити змінні середовища:

```
export PGSQL15=/var/lib/postgresql/15.3/
export PATH=$PGSQL15/bin:$PATH
export LD_LIBRARY_PATH=$PGSQL15/lib
export PGDATA=$PGSQL15/data
```

Крок 13. Ініціалізувати системні файли каталогу бази даних:

```
initdb
```

Крок 14. Запустити сервер СКБД *PostgreSQL*:

```
pg_ctl start
```

Крок 15. Перевірити з’єднання програми-клієнта із сервером СКБД *PostgreSQL:*

```
psql
```

Крок 16. Зупинити сервер СКБД *PostgreSQL*:

```
pg_ctl stop
```

1.3.5 Керування обмеженням використання ресурсів ОС

Якщо сервер з *ОС Linux* використовується багатьма користувачами, тоді можливі ситуації, коли програми одного користувача помилково почнуть активно використовувати ресурси сервера, наприклад, оперативну пам'ять або потужність процесора, обмежуючи роботу інших користувачів. Тому важливо контролювати всі процеси, обмежуючи дії користувачів. Таке керування може бути застосовано на трьох рівнях:

- глобальний рівень – для всіх користувачів одночасно;
- груповий рівень – для окремих груп користувачів;
- користувальницький рівень – для окремих користувачів.

Для перегляду значення поточних обмежень та для тимчасової зміни цих значень використовується команда:

```
ulimit [options] [user-name]
```

Опція *user-name* визначає ім’я користувача, для якого описується обмеження. Якщо *user-name* не визначено, тоді параметри визначаються для поточного користувача.

Деякі приклади значень для *options*:

– a – отримати значення всіх параметрів;

– *f* – максимальний розмір файлу (Кб);

– *n* – максимальна кількість відкритих файлових дескрипторів;

– *t* – максимальний час процесора (хвилини);

– *u* – максимальна кількість процесів;

– *v* – максимальний розмір віртуальної пам’яті (Кб).

На рисунку 21 наведено результат виконання команди *ulimit –a*

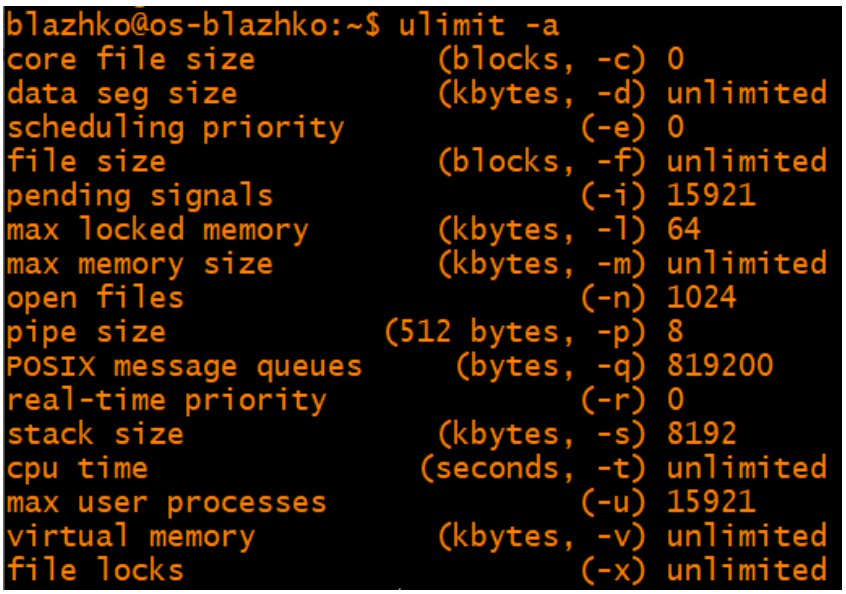


Рис. 21 – Результат виконання команди *ulimit -a*

В таблиці 1 наведено приклади команд *ulimit.*

Таблиця 1 – Приклади команд *ulimit*

Приклад команди	Приклад операції, яка порушує обмеження	Приклад повідомлення про помилку щодо порушення обмеження
ulimit -f 10	*cp /bin/bash .*	*File size limit exceeded (core dumped)*
ulimit -n 2	*ls*	*-bash: start_pipeline: pgrp pipe: Too many open files*
ulimit -t 1	*while [true]; do ((x++)); done*	*Connection to 127.0.0.1 closed.*
ulimit -u 1	*ls*	*-bash: fork: retry: No child processes*
ulimit -v 1	*ls*	*Segmentation fault*

На рисунку 22 наведено приклади виконання трех команд встановлення обмежень, приклади операції, які порушують обмеження та приклад повідомлень про помилку щодо порушення обмежень.

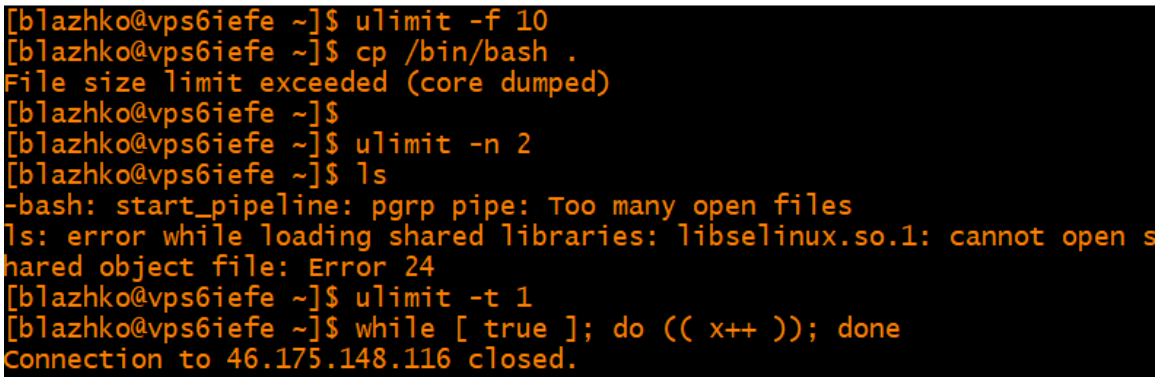


Рис. 22 – Приклади виконання трех команд встановлення обмежень

Для повернення значень змінених обмежень параметрів до початкових вже неможливо використати команду *ulimit*, бо на таку операцію ОС вже видасть повідомленнґя про помилку:

cannot modify limit: Operation not permitted

Початкові значення будуть повернені лише після виходу із псевдотерміналу та повторного входу до ОС.

Для постійної зміни обмежень використовується файл */etc/security/limits.conf*, в якому правила зберігаються у вигляді рядків:

<domain > <type> <item> <value>

- *domain* – рівень застосування обмеження:
    - ім’я групи користувачів, перед якою вказується символ @
    - ім’я користувача;
- *type* – тип обмеження:
    - *hard* – значення ресурсу може бути встановлено лише адміністратором ОС;
    - *soft* – значення ресурсу може бути встановлено звичайним користувачем, але не може перевищити значення *hard*-типу;
- *item* – назва ресурсу, який обмежується, наприклад:
    - *fsize* – максимальний розмір файлу (КБ);
    - *nofile* – максимальна кількість відкритих файлових дескрипторів;
    - *cpu* – максимальний час процесора (хвилини);
    - *nproc* – максимальна кількість процесів;
    - *memlock* – максимальний розмір віртуальної пам’яті (Кб).
- *value* – чисельне значення.

Для того, щоб зміни набули чинності, потрібне перезавантаження ОС.

Приклади налаштування файлу присутні в самому файлі.

2 Завдання до лабораторної роботи

На відміну від попередніх лабораторних робіт у цій лабораторній роботі завдання виконуються на вашому локальному комп'ютері.

Виконати наступні дії з підготовки до виконання завдань роботи:

1) використовуючи команди *Bash*, перейти до каталогу *Git*-репозиторія, який був створений ще лабораторній роботі №4;

2) за необхідністю оновити зміст *Git*-репозиторія змістом з *GitHub*-репозиторію командою *pull;*

3) створити нову *Git*-гілку з назвою «*Laboratory-work-14»*;

4) перейти до роботи зі створеною гілкою;

5) створити каталог з назвою «*Laboratory-work-14»*;

6) перейти до каталогу «*Laboratory-work-14»;*

7) в каталозі «*Laboratory-work-14»* створити файл *README.md* та додати до файлу рядок тексту із темою лабораторної роботи *«Основи керування віртуальними ОС»* як заголовок 2-го рівня *Markdown*-форматування;

8) виконати операції з оновлення *GitHub*-репозиторію змінами *Git*-репозиторія через послідовність *Git*-команд *add, commit* із коментарем «*Changed by Local Git*» та *push*;

9) на веб-сервісі *GitHub* перейти до створеної гілки «*Laboratory-work-14»;*

10) перейти до каталогу «*Laboratory-work-14»* та розпочати процес редагування файлу *README.md*

11) в подальшому за результатами рішень кожного наступного розділу завдань до файлу *README.md* додавати:

- рядки як заголовки 3-го рівня Markdown-форматування з назвами розділу;
- знімки екранів, які демонструють успішність виконання пункту завдання;
- підписи під кожним знімком екрану із повним описом пункту завдання;
- створення знімків екрану роботи віртуальної машини можна зробити через пункт меню *View -> Take Screenshort…*

Примітка про особливості виконання роботи на комп'ютерах із *Apple Silicon ARM*-процесорами (*M1/M2*):

– на сторінці https://www.virtualbox.org/wiki/Downloads є коментар "*Developer preview for macOS/Arm64 (M1/M2) hosts*", але відгуки користувачів вказують на те, що запропонований інсталяційний пакунок, можливо, працює для *Intel*-процесорів, але для *ARM*-процесорів не працює,

– на відео-запису https://www.youtube.com/watch?v=tumi5TT1Xxs на початок березня 2023 року стверджується, що з використанням менеджеру *Homebrew (The Missing Package*

Manager for macOS) інсталяція можлива, хоча на початок червня 2023 року в описі на сторінках https://formulae.brew.sh/cask/virtualbox, https://github.com/Homebrew/homebrew-cask/blob/master/Casks/virtualbox.rb є лише посилання на *Intel*-процесор;

– враховуючи попередні пункти, студентам-власникам лише комп'ютеру із *Apple Silicon ARM*-процесором, пропонується самостійно розібратися із встановленням програм, які дозволять запустити будь-яку ОС *Linux*, наприклад, розібратися із такими програмами:

- https://www.vmware.com/nl/products/fusion/fusion-evaluation.html
- https://www.parallels.com/nl/products/desktop/

– будь-який успішний запуск ОС *Linux* на комп'ютері із *Apple Silicon ARM*-процесорами визначить успішність виконання завдань 2.1 та 2.2.

2.1 Створення віртуальної машини через графічний інтерфейс

2.1.1 Встановити програмне забезпечення *Oracle VM Virtual Box*

2.1.2 Отримати з інтернету *ISO*-образ для 32-бітної ОС *Server Linux Ubuntu* 14.04.6.

2.1.3 Створити віртуальну машину, використовуючи графічний інтерфейс та враховуючи наступні значення параметрів:

– назва = *«Linux of Surname»*, де *Surname* – ваше прізвище транслітерацією, наприклад *«Linux of Blazhko»*;

– файл *ISO*-образу = *ubuntu-14.04.6-server-i386.iso*;

– розмір оперативної пам'яті (Мб) = *10 * group + var*, де *group* – номер вашої групи, *var* – номер вашого варіанту, наприклад, *10 * 221 + 5 = 2215 Мб;*

– кількість ядер процесору = 1;

– розмір жорсткого диску (Мб) = *40 * group*, наприклад, *40 * 221 = 8840 Мб = 8,84 Гб;*

– правило *Port Forwarding* на першу мережеву карту, яке буде перенаправляти мережеві пакети для з'єднання через *SSH*-протокол із *guest*-ОС (*IP-address=10.0.2.15, SSH-port=22*) на *IP-адресу=127.0.0.1* з використанням *SSH-port=1234.*

2.1.4 Запустити віртуальну машину, створену у пункті 2.1.3, та виконати інсталяцію віртуальної ОС, враховуючи параметри:

– *hostname = os-surname,* наприклад, *os-blazhko*;

– *user account = surname,* наприклад, *blazhko*;

– *time zone = Kiev*;

– *Software = OpenSSH server;*

2.1.5 Після інсталяції та перезапуску віртуальної машини виконати вхід до віртуальної ОС, використовуючи створений обліковий запис.

2.1.6 Повторити вхід до віртуальної машини через *SSH*-протокол.

2.2 Створення віртуальної машини через інтерфейс командного рядку

2.2.1 Створити віртуальну машину, використовуючи інтерфейс командного рядку та враховуючи наступні значення параметрів:

– назва = *«Linux of Surname 2»*, де *Surname* – ваше прізвище транслітерацією, наприклад *«Linux of Blazhko»*;

– розмір оперативної пам'яті (Мб) = *10 * group + var*, де *group* – номер вашої групи, var – номер вашого варіанту, наприклад, *10 * 221 + 5 = 2215 Мб;*

– кількість ядер процесору = 1;

– розмір оперативної пам'яті графічної карти (Мб) = 100 + *group2,* останні дві цифри номеру групи, наприклад, *100 + 21 = 121 Мб*;

– перша мережева карта з конфігурацією *NAT;*

– правило *Port Forwarding* на першу мережеву карту, яке буде перенаправляти мережеві пакети для з'єднання через *SSH*-протокол із *guest*-ОС (*IP-address=10.0.2.15, SSH-port=22*) на *IP-адресу=127.0.0.1* з використанням *SSH-port=1234;*

– контролер *IntelAHCI* з інтерфейсом *sata*-типу за назвою *SurnameSATA*, *Surname* – ваше прізвище транслітерацією, наприклад *«BlazhkoSATA»*;

– контролер *PIIX4* з інтерфейсом *ide*-типу за назвою *SurnameIDE*, де *Surname* – ваше прізвище транслітерацією, наприклад *«BlazhkoIDE»*;

– віртуальне сховище даних, розміщене у файлі з назвою *DiskSurname.vdi,* наприклад, *DiskBlazhko.vdi,* та розміром (Мб) = *40 * group*, наприклад, *40 * 221 = 8840 Мб;*

– *ISO*-файл *ubuntu-14.04.6-server-i386.iso* з інсталяцією ОС *Linux*

2.2.2 Запустити віртуальну машину, створену у пункті 2.2.1, використовуючи інтерфейс командного рядку.

2.2.3 Перебуваючи на першому екрані процесу інсталяції віртуальної ОС, призупинити роботу віртуальної машини.

2.2.4 Повторно запустити призупинену віртуальну машину, створену у пункті 2.2.1.

2.2.5 Зупинити віртуальну машину, створену у пункті 2.2.1.

2.2.6 Зняти з реєстрації віртуальну машину та видалити всі пов'язані з нею файли.

2.2.7 Зберегти всі команди керування віртуальною машиною, які було виконано у попередніх пунктах, у файлі *CreateVMSurname.sh*, де *Surname* – ваше прізвище транслітерацією, наприклад *«CreateVMBlazhko.sh»*;

2.3 Створення групи користувача та облікового запису користувача ОС *Linux*

2.3.1 Встановити з'єднання із віртуальною машиною, яка створена у розділі 2.1., через *SSH*-команду.

2.3.2 Створити групу користувача, назва якої співпадає з назвою вашої студентської групи транслітерацією, наприклад, *AI211*;

2.3.3 З використанням не інтерактивної команди створити обліковий запис користувача з урахуванням того, що:

– ім`я користувача співпадає з об'єднаними вашим прізвищем та іменем транслітерацією, наприклад, *blazhko_oleksandr*;

– каталог користувача співпадає з вашим ім'ям;

– шлях до оболонки командного рядка = */bin/bash*

– користувач входить до раніше створеної групи.

2.3.4 В окремому псевдотерміналі увійти з'єднатися з віртуальною машиною під іменем створеного користувача. Вийти із псевдотерміналу.

2.3.5 Видалити створений обліковий запис користувача.

2.3.6 Видалити створену групу.

2.4 Встановлення СКБД *PostgreSQL* та налаштування роботи

2.4.1 Використовуючи команду керування програмними пакунками, наприклад, *apt*, встановити програмний пакунок СКБД *PostgreSQL*

2.4.2 Від імені користувача *postgres* створити БД *Surname*, де *Surname* – ваше прізвище транслітерацією.

2.4.3 Від імені користувача *postgres* створити користувача СКБД з іменем *Surname*, де *Surname* – ваше прізвище транслітерацією.

2.4.4 З'єднатися з СКБД *PostgreSQL,* використовуючи створеного користувача та БД.

2.4.5 Від імені користувача *postgres* зупинити сервер СКБД.

2.4.6 Видалити програмний пакунок СКБД *PostgreSQL.*

2.5 Встановлення останньої версії СКБД *PostgreSQL* із сирцевих кодів

2.5.1 Встановити програмні пакунки, від яких залежить побудова СКБД *PostgreSQL* версії 15.3 із сирцевих кодів

2.5.2 Виконати процес побудови СКБД *PostgreSQL* версії 15.3 із сирцевих кодів та виконати інсталяцію програм серверу.

2.5.3 Запустити сервер СКБД *PostgreSQL* версії 15.3.

2.5.4 Перевірити з'єднання програми-клієнта із сервером СКБД *PostgreSQL,* використовуючи параметри зазамовчуванням.

2.5.5 Зупинити сервер СКБД *PostgreSQL.*

2.6 Керування обмеженням використання ресурсів ОС

2.6.1 Встановити з'єднання із віртуальною машиною, яка створена у розділі 2.1., через *SSH*-команду від імені користовувача, створеного під час інсталяції віртуальної ОС.

2.6.2 Перевірити роботу команди *ulimit* за всіма прикладами з таблиці 1 з підрозділу 1.3.5, показавши на екрані:

- приклад кожної команд;
- приклад операції, яка порушує обмеження;
- приклад повідомлення про помилку щодо порушення обмеження.

2.7 Підготовка процесу *Code Review* для надання рішень завдань лабораторної роботи на перевірку викладачем

Примітка: Рішення на завдання 2.1-2.2 можуть бути надані під час *Online*-заняття для отримання відповідних балів. За межами *Online*-заняття виконуються всі рішення.

На веб-сервісі *GitHub* зафіксувати зміни у файлі *README.md*.

Скопіювати файли, які було створено у попередніх розділах завдань, в каталог «*Laboratory-work-14*» *Git*-репозиторію.

Оновити *Git*-репозиторій змінами нової гілки «*Laboratory-work-14*» з *GitHub*-репозиторію.

Оновити *GitHub*-репозиторій змінами нової гілки «*Laboratory-work-14*» *Git*-репозиторію. Виконати запит *Pull Request*.

Примітка: Увага! Не натискайте кнопку «Merge pull request»!

Це повинен зробити лише рецензент, який є вашим викладачем!

Може бути виконано два запити *Pull Request*: під час *Online*-заняття та за межами *Online*-заняття.

Якщо запит *Pull Request* було зроблено під час *Online*-заняття, тоді рецензент-викладач перегляне рішення, надасть оцінку та закриє *Pull Request* без операції *Merge*.

Коли буде зроблено остаточний запит *Pull Request*, тоді рецензент-викладач перегляне ваше рішення та виконає злиття нової гілки та основної гілки через операцію *Merge*. Якщо рецензент знайде помилки, він повідомить про це у коментарях, які з'являться на сторінці *Pull request.*

2.8 Оцінка результатів виконання завдань лабораторної роботи

Оцінка	Умови
+2 бали	під час *Online*-заняття виконано правильні рішення завдань 2.1-2.2 або на наступному *Online*-занятті чи на консультації отримано правильну відповідь на два запитання, які стосуються рішень
+2 бали	1) всі рішення роботи відповідають завданням 2) *Pull Request* представлено не пізніше найближчої консультації після офіційного заняття із захисту лабораторної роботи
-0.5 балів за кожну омилку	в рішенні є помилка, про яку вказано в *Code Review*
-1 бал	*Pull Request* представлено пізніше часу завершення найближчої консультації після офіційного заняття із захисту лабораторної роботи за кожний тиждень запізнення
+2 бали	на наступному *Online*-занятті або на консультації отримано правильну відповідь на два запитання, які стосуються рішень

Література

1. Блажко О.А. Відео-запис лекції «*Основи керування віртуальними ОС*». URL: https://youtu.be/eO_PGNpSkwo