МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова

П. Й. ГУЧЕК, О. І. ЛИТВИНЕНКО

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з курсу «ОПЕРАЦІЙНІ СИСТЕМИ»

Рекомендовано Методичною радою НУК

ВИДАВНИЦТВО
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
КОРАБЛЕБУДУВАННЯ
ІМ. АДМІРАЛА МАКАРОВА

2022

УДК 681.3.06
Г97

Автори:
П. Й. Гучек, д-р техн. наук, доцент, завідувач кафедри інформаційних технологій та фізико-математичних дисциплін Херсонського навчально-наукового інституту НУК ім. адм. Макарова;
О. І. Литвиненко, канд. техн. наук, доцент кафедри інформаційних технологій та фізико-математичних дисциплін Херсонського навчально-наукового інституту НУК ім. адм. Макарова

Рецензенти:
О. М. Дудченко, канд. техн. наук, професор;
М. Б. Літвінова, д-р пед. наук, канд. фіз.-мат. наук, професор

Рекомендовано Методичною радою НУК

Гучек П. Й.

Г97 Методичні вказівки до виконання лабораторних робіт з курсу «Операційні системи» / П. Й. Гучек, О. І. Литвиненко. – Миколаїв : НУК, 2022. – 114 с.

У вказівках подано завдання до лабораторних робіт та приклади їх виконання.

Призначені для студентів Херсонської філії НУК, які навчаються за спеціальністю 121 «Інженерія програмного забезпечення», а також для студентів усіх інших споріднених спеціальностей, що вивчають основи операційних систем.

УДК 681.3.06

ЗМІСТ

ВСТУП

Дисципліна «Операційні системи» призначена для формування у студентів систематизованого уявлення про основні принципи функціонування операційних систем, про концепції, структури і механізми, що лежать в основі роботи основних функцій сучасних операційних систем, їх характеристик і про сучасні напрямки їх розвитку. У лабораторних роботах розглядаються основні принципи функціонування операційної системи LINUX, можливості файлової системи і функцій по обробці та управління даними, методи динамічного породження процесів і способи передачі інформації між взаємодіючими процесами.

Зміст лабораторної роботи студентів з дисципліни визначається навчальною програмою дисципліни та робочою навчальною програмою вивчення дисципліни.

Навчально-методичні засоби лабораторної роботи студентів:

– основна література (підручник, конспект лекцій, навчальні та методичні посібники);

– додаткова література (наукова, фахова, періодична);

– методичні матеріали.

Вимоги до лабораторної роботи студента:

– робота має бути виконана особисто студентом або групою студентів, де кожен її член самостійно виконує свою частку колективної роботи;

– робота повинна являти собою закінчену розробку (або її етап), де розкриваються й аналізуються актуальні проблеми з певної теми або її окремих аспектів;

– робота має демонструвати достатню компетентність автора (авторів) у розкритті питань, що досліджуються;

– робота повинна мати навчальну, наукову й (або) практичну спрямованість і значимість, містити певні елементи новизни (при виконанні науково-дослідної роботи).

Оформлення звітів з лабораторної роботи студентів здійснюється відповідно до вимог, розроблених кафедрою, та

інших нормативних документів, що стосуються виконання та оформлення наукових, навчально-методичних та інших робіт.

Формами контролю студентів за якістю оволодіння навчальним матеріалом є: самоконтроль за допомогою контрольно-тестових завдань та контроль з боку викладача, який здійснюється за допомогою методів поточного і підсумкового контролю.

Методами поточного контролю є:

– усне опитування студентів на лабораторних заняттях;

– перевірка практичних завдань, виконаних студентами індивідуально;

– перевірка рефератів та організація їх презентацій;

– співбесіда.

Формами підсумкового контролю є:

– проведення контрольних робіт в аудиторії (за тематичними модулями);

– екзамен (залік) за передбаченими програмою курсу питаннями та практичними завданнями.

Облік успішності студентів з виконання лабораторної роботи здійснюють викладачі у журналах обліку успішності.

Метою дисципліни є формування у студентів здатностей:

– володіти знаннями щодо принципів роботи операційних систем;

– мати навички керування ресурсами обчислювальної системи, взаємодії з прикладним програмним забезпеченням, а також уміти обґрунтовано вибрати операційну систему для вирішення певних завдань і грамотно її налаштовувати;

– уміти керувати розподіленими ресурсами обчислювальної системи.

У методичні вказівки відповідно до курсу входить чотирнадцять лабораторних робіт, призначених для прищеплення студентам навичок:

– здійснювати інсталяцію сучасних операційних систем Linux і Windows, у тому числі у віртуалізованих середовищах;

– виконувати базові настроювання операційних систем і вирішувати задачі їх адміністрування;

– використовуючи системні засоби, розробляти сценарії для автоматизації задач адміністрування;

– формулювати вимоги до операційної системи для вирішення певних прикладних завдань.

Кожна лабораторна робота складається з коротких теоретичних відомостей, завдань для самостійного виконання та контрольних питань.

Робота № 1. Робота з операційними системами. Використання віртуальної машини

***Мета роботи**:* ознайомлення з перевагами і недоліками віртуальних машин, а також вивчення можливостей менеджера віртуальних машин Oracle VirtualBox.

Короткі теоретичні відомості

Віртуальна машина – це повністю ізольований програмний контейнер, здатний виконувати власну операційну систему і додатки, як фізичний комп'ютер. Віртуальна машина працює абсолютно так само, як фізичний комп'ютер, і містить власні віртуальні (тобто програмні) ЦП, ОЗУ, жорсткий диск і мережеву інтерфейсну карту.

По суті, віртуальна машина – це програма, яка запускається з операційної системи. Програма емулює реальну машину. На віртуальні машини, як і на реальні, можна ставити операційні системи. У неї є BIOS, відведене місце на жорсткому диску, мережеві адаптери для з'єднання з реальною машиною, мережевими ресурсами або іншими віртуальними машинами.

Емуляція (англ. *emulation*) – комплекс програмних і апаратних засобів, призначених для копіювання (або емуляції) функцій однієї обчислювальної системи на іншу, відмінну від першої, таким чином, щоб поведінка якомога ближче відповідала поведінці оригінальної системи.

Переваги віртуальних машин

Перед можливістю установки декількох хостових операційних систем на один комп'ютер з їх роздільним завантаженням, віртуальні машини мають наступні незаперечні переваги:

• можливість працювати одночасно в декількох системах, здійснювати мережну взаємодію між ними;

• можливість зробити «знімок» поточного стану системи і вмісту дисків одним кліком миші, а потім протягом дуже короткого проміжку часу повернутися в початковий стан;

• простота створення резервної копії операційної системи (не треба створювати ніяких образів диска, всього

лише потрібно скопіювати папку з файлами віртуальної машини);

• можливість мати на одному комп'ютері необмежену кількість віртуальних машин з абсолютно різними операційними системами та їх станами;

• відсутність необхідності перезавантаження для перемикання в іншу операційну систему.

Недоліки віртуальних машин

Незважаючи на всі переваги, віртуальні машини також мають і свої недоліки:

• потреба в наявності достатніх апаратних ресурсів для функціонування декількох операційних систем одночасно;

• операційна система працює трохи повільніше в віртуальній машині;

• існують методи визначення того, що програма запущена у віртуальній машині (в більшості випадків виробники систем віртуалізації самі надають таку можливість). Віруписьменники і розповсюджувачі шкідливого програмного забезпечення, звичайно ж, в курсі цих методів і останнім часом у своїх програмах функції виявлення факту запуску в віртуальній машині, при цьому ніякої шкоди програмному забезпеченню гостьовій системі не заподіює;

• різні платформи віртуалізації поки не підтримують повну віртуалізацію всього апаратного забезпечення та інтерфейсів (USB, CD-ROM і т. п.).

Завантаження програми VirtualBox

VirtualBox – дуже простий, потужний і безкоштовний інструмент для віртуалізації, що розвивається завдяки підтримці корпорації Oracle. Він поширюється безкоштовно, з відкритим вихідним кодом. VirtualBox дозволяє встановлювати як гостьову практично будь-яку сучасну операційну систему, таку як Windows, MacOS або будь-яку з численних представників сімейства Linux.

Програму можна завантажити з сайту розробника:

https://www.virtualbox.org/

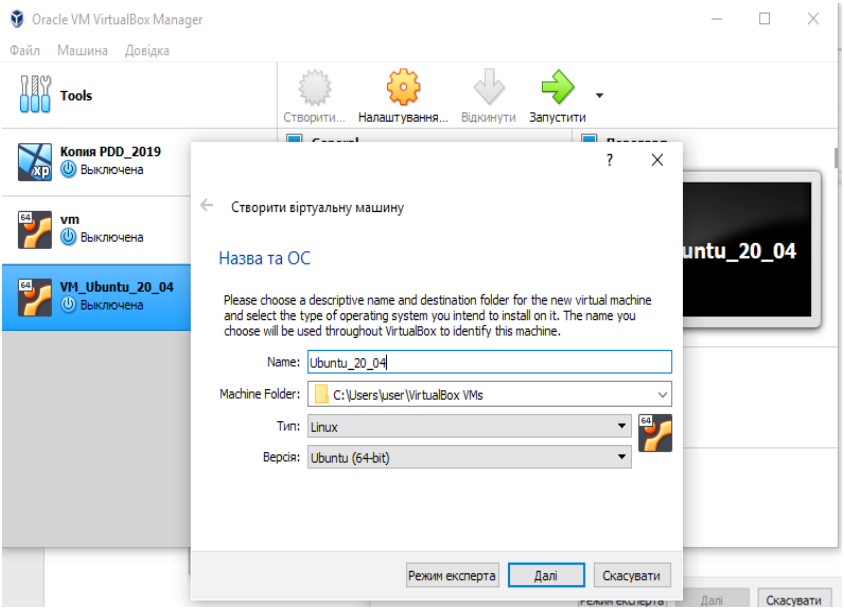
Хід роботи

Створення віртуальної машини

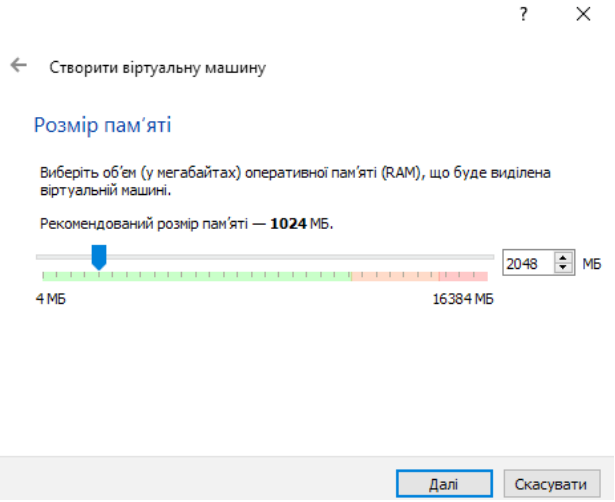
Створимо, наприклад, віртуальну машину для установки операційної системи Linux версії Ubuntu. Для неї виділимо оперативної пам'яті (2048 Мбайт), пам'яті відеокарти (64 Мбайт), задіємо CD ROM і вкажемо необхідність використання Flash-диска.

Після запуску програми VirtualBox відкривається її вікно. Клацанням миші по кнопці «Створити» починається послідовність кроків по створенню нової віртуальної машини:

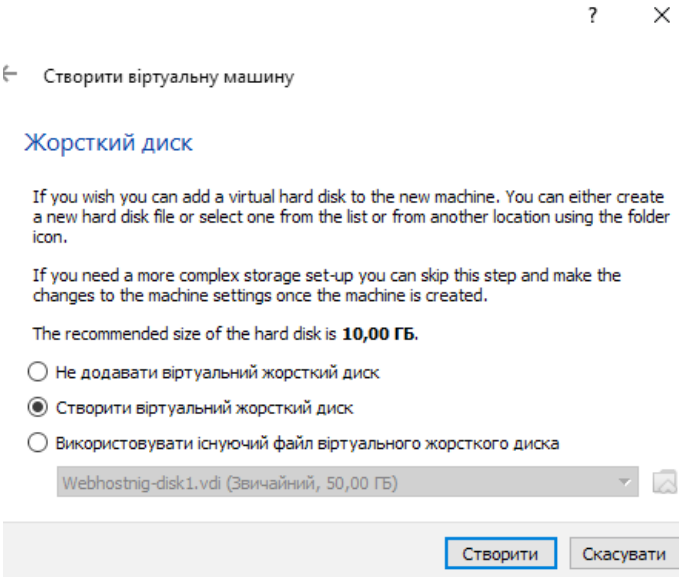
- Задаємо ім'я віртуальної машини (відповідно до завдання кращим буде ім'я Ubuntu), тип операційної системи і версію:



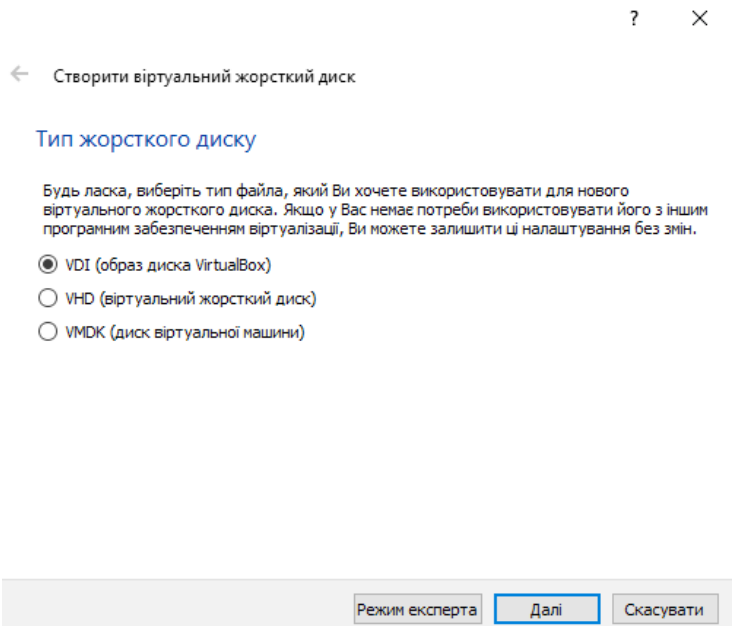
- Клацнемо мишею по кнопці «Далі». У цьому кроці задаємо об'єм пам'яті для віртуальної машини. За умовами завдання – це 2048 Мбайт



- Клацнемо мишею по кнопці «Далі». Зараз потрібно вказати, який обсяг пам'яті на жорсткому диску може використовувати віртуальна машина. У вас є «створити новий жорсткий диск» або «використовувати існуючий жорсткий диск», залишаємо галочку навпроти «Створити віртуальний жорсткий диск:



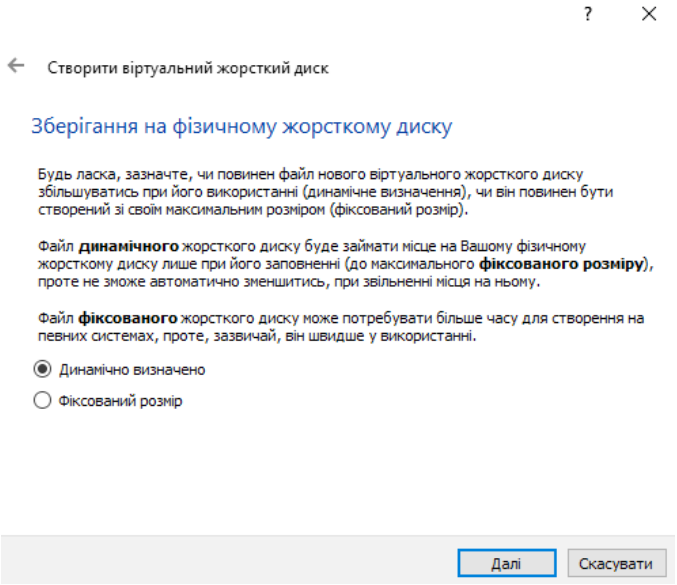
- Клацнемо мишею по кнопці «Створити» і виберемо тип жорсткого диску:



Клацнемо мишею по кнопці «Далі».

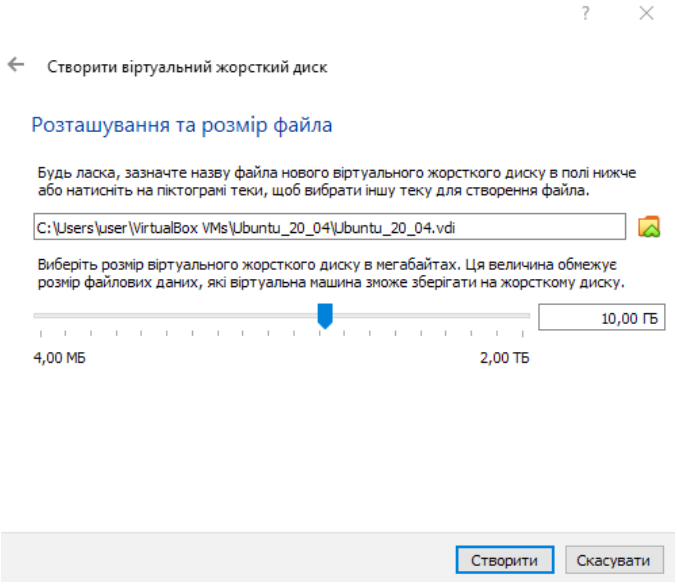
- В результаті на фізичному жорсткому диску буде записано спеціальний файл, який буде використовуватися як віртуальний диск. Варіантів для створення віртуального жорсткого диску два:
  - фіксований розмір, на фізичному жорсткому диску відразу буде записаний файл відповідних розмірів;
  - динамічний образ, який розширюється на фізичному жорсткому диску.

В даному випадку виберемо файл динамічного жорсткого диску:

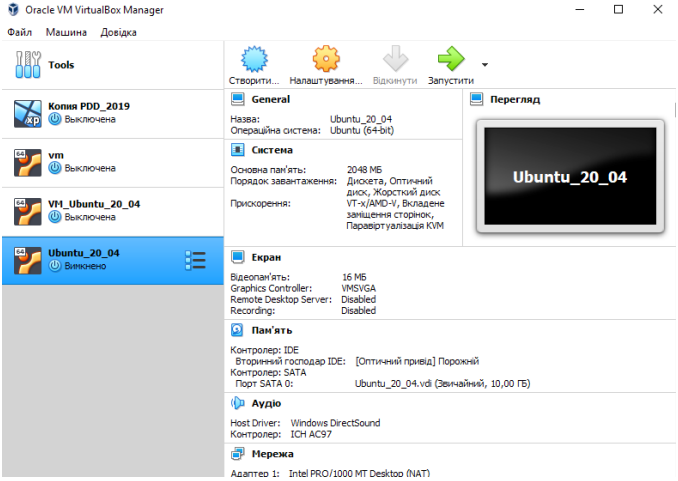


Клацнемо мишею по кнопці «Далі».

- Зазначимо розташування та розмір жорсткого диска, який буде використаний як віртуальний диск:



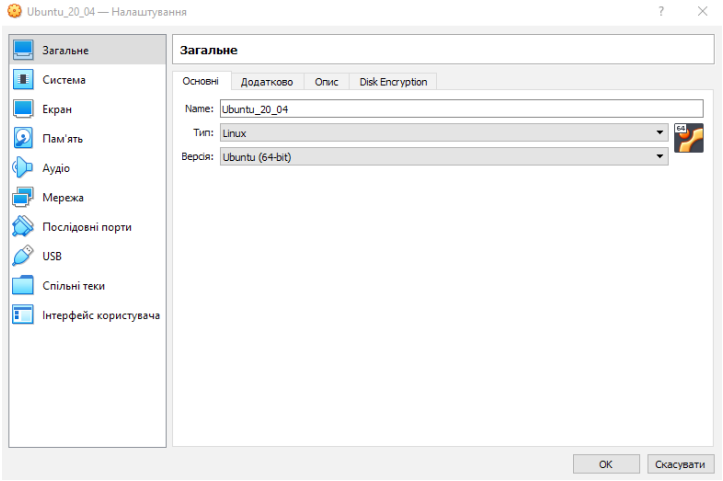
Клацнемо мишею по кнопці «Створити». Віртуальна машина готова:



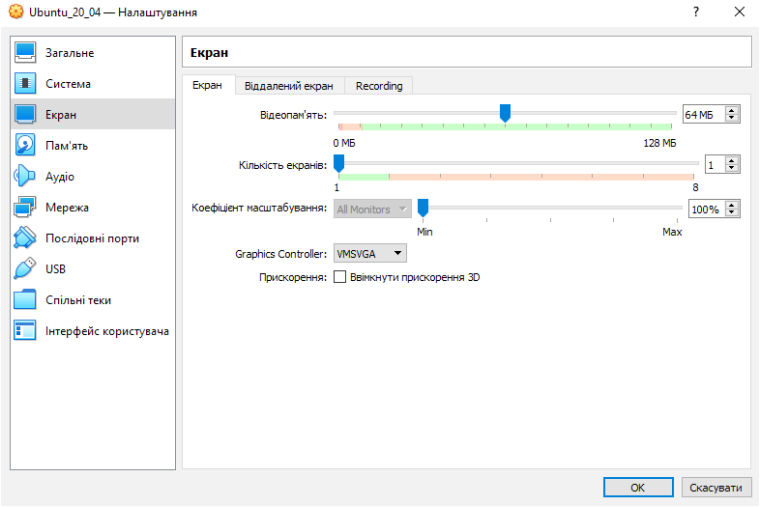
Налаштування віртуальної машини на базі VirtualBox

Після установки програми VirtualBox і створення віртуальної машини необхідна її додаткова настройка для установки зазначених параметрів.

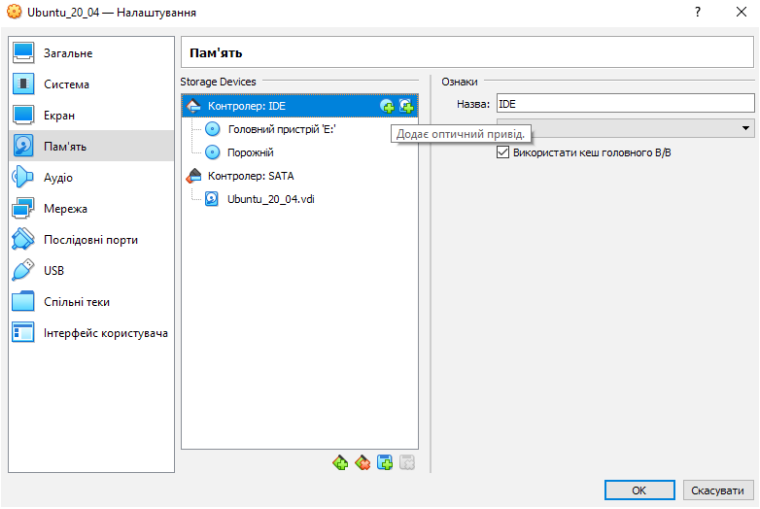
- Почнемо настройку з клацання мишею по кнопці «Налаштування». Відкриється вікно діалогу «Налаштування»:



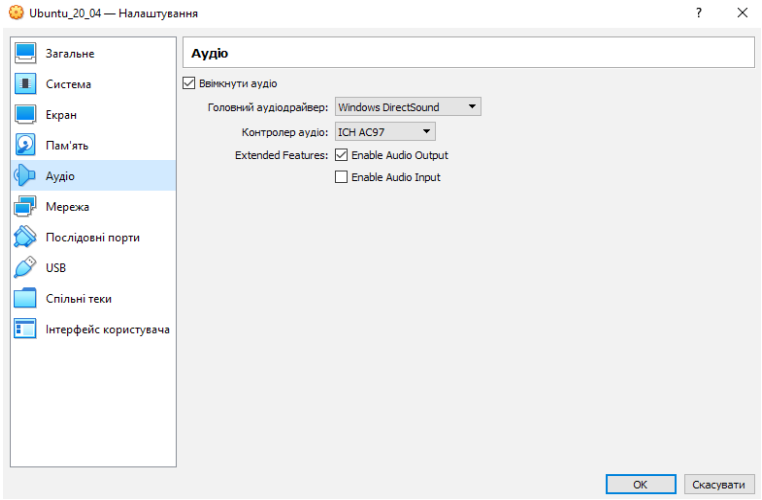
У розділі «Екран» задаємо необхідний об’єм відеопам’яті (64 Мбайт):



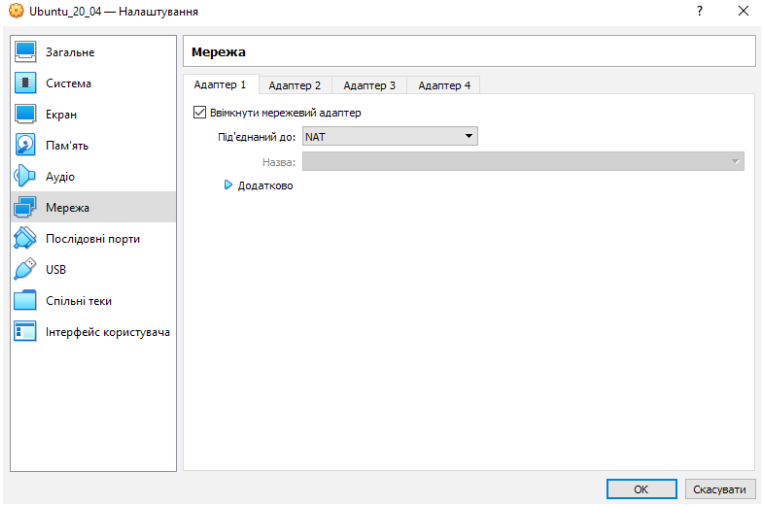
- Вибираємо в меню розділ «Пам’ять» і встановлюємо для нього параметри «Підключити CD / DVD» або «Привід хоста ‘E:’» за допомогою натискання кнопки «Додати привід оптичних дисків»:



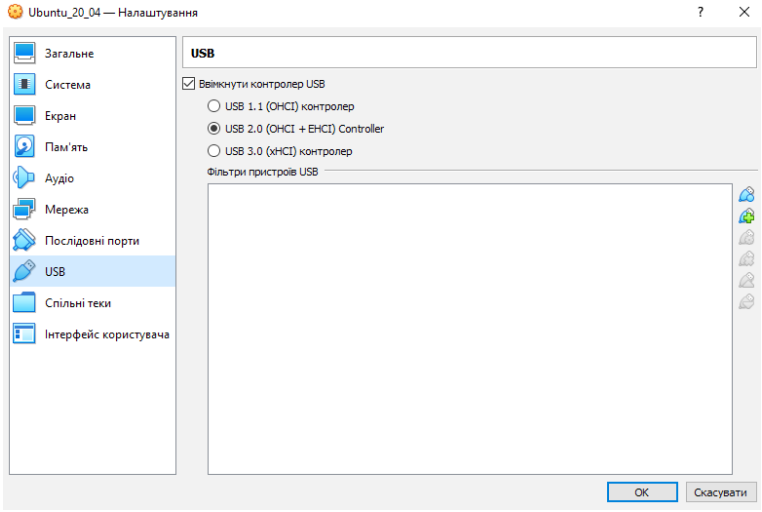
- При бажанні можна включити аудіопристрій і вказати для нього драйвер, наприклад:



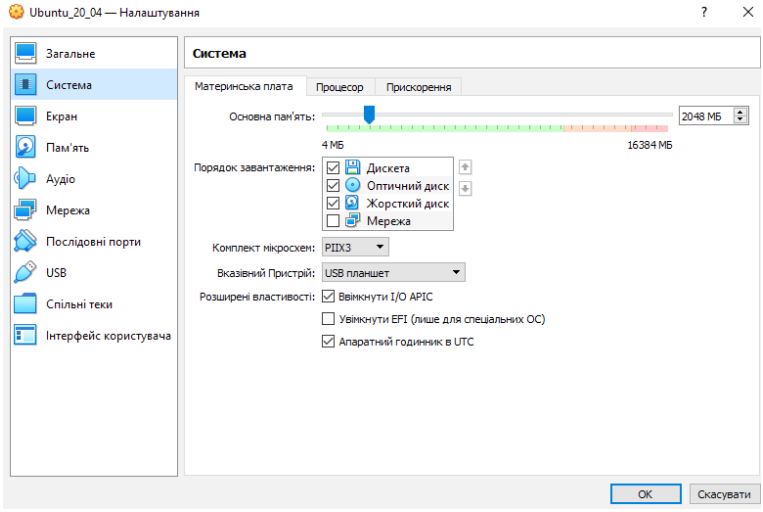
- Перевіряємо параметри мережевих пристроїв віртуальної машини:



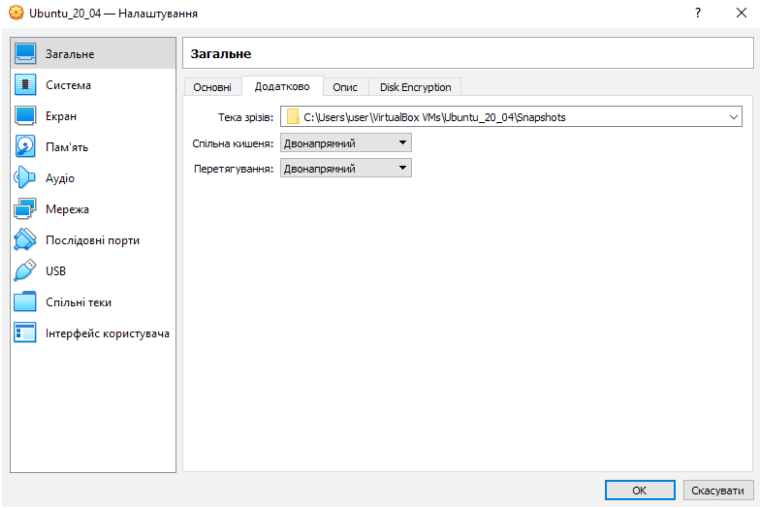
- Тепер включаємо контролер USB:



- Вибираємо в меню розділ «Система» і картку «Материнська плата». Встановлюємо порядок звернення до пристроїв для завантаження:



▪ У розділі меню «Загальне» у вкладці «Додатково» включаємо / вимикаємо загальний буфер обміну. Для прикладу він включений. Можна вказати й інші варіанти використання буфера обміну.



Основні параметри перевірені і задані. Клацнемо мишею по кнопці «ОК».

Запуск віртуальної машини VirtualBox

Запускається налаштована віртуальна машина клацанням миші по кнопці «Запустити». В дисковод необхідно поставити завантажувальний диск. Після цього на тлі вікна VirtualBox відкривається вікно віртуальної машини.

Передача управління пристроями «клавіатура» і «миша» з гостьової операційної системи в основну і назад виконується натисканням правої клавіші <Ctrl>.

Установка Linux Ubuntu на віртуальну машину Oracle VM VirtualBox

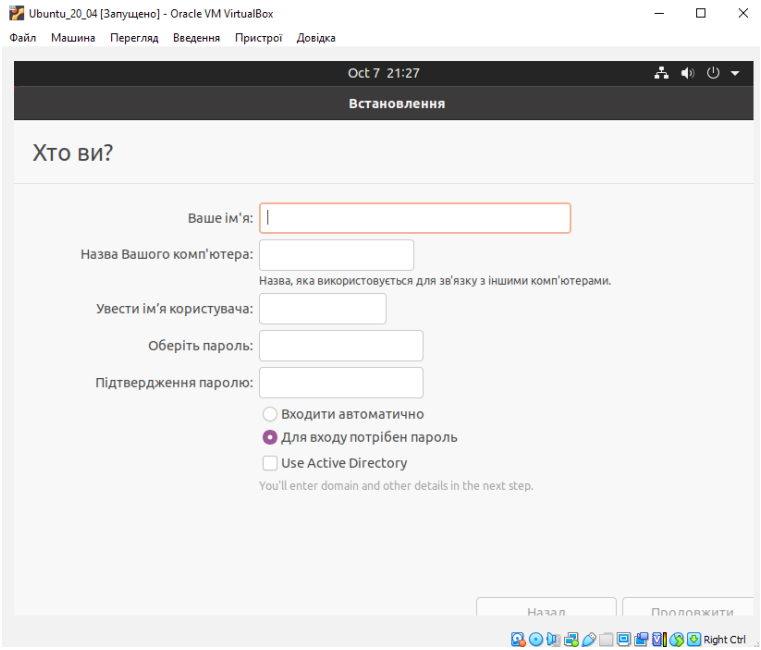
Для установки операційної системи на віртуальній машині потрібно завантаження з інсталяційного диска. У середовищі VirtualBox є можливість виконання завантаження не тільки зі стандартних пристроїв (CD/DVD-привід,

флешка, мережа ...), але і з використанням віртуального приводу, створюваного на основі образу завантажувального диска. Зазвичай дистрибутиви Linux поширюються у вигляді файлів образів у форматі ISO-9660 (файлів з розширенням iso) і VirtualBox дозволяє обійтися без запису образу на компакт диск, а просто підключити такий файл безпосередньо до віртуальної машини в якості віртуального приводу з установленим носієм на основі вмісту iso-образу. При першому запуску віртуальної машини, коли ще немає встановленої гостьовоъ операційної системи, VirtualBox запропонує вибрати пристрій завантаження.

Замість фізичного приводу можна вибрати файл образу, наприклад ubuntu-20.04.3-desktop-amd64.iso, який буде підключений як віртуальний пристрій, що містить інсталяційний CD/DVD диск Ubuntu 20.04. При натисканні на кнопку Продовжити виконається завантаження з віртуального приводу і почнеться установка гостьової операційної системи (Ubuntu).

Процес установки гостьової ОС нічим не відрізняється від установки на реальній машині. Можна вибрати мову для встановлюваної системи (зазвичай Українська), часовий пояс, розкладку клавіатури і т. п. Більшість параметрів можна залишити за замовчуванням, у тому числі і тип установки.

В процесі установки необхідно задати ім’я комп’ютера, користувача пароль і режим входу в систему:



Подальша установка Ubuntu виконується без будь-якого втручання користувача і завершується пропозиціями перезавантажити комп'ютер. У порівнянні з установкою системи на реальному комп'ютерному обладнанні, установка на віртуальній машині виконується повільніше, що цілком очікувано. Ступінь зниження продуктивності в основному, залежить від швидкодії обладнання реального комп'ютера.

При першому завантаженні знову встановленої операційної системи, диспетчер VirtualBox автоматично відключить віртуальний привід на основі образу диска з дистрибутивом Ubuntu, завантаження буде виконана з віртуального жорсткого диска і по її завершенню, на екрані відобразиться запрошення до входу в систему.

Знайомство з операційною системою Linux

Поняття login і password

Операційна система Linux є багатокористувацькою операційною системою. Для забезпечення безпечної роботи

користувачів і цілісності системи доступ до неї повинен бути санкціонований. Для кожного користувача, якому дозволено вхід у систему, заводиться спеціальне реєстраційне ім'я ***username*** або ***login*** і зберігається спеціальний пароль ***password***, що відповідає цьому імені.

Вхід у систему

Якщо в системі встановлена графічна оболонка поряд зі звичайними алфавітно-цифровими терміналами, найкраще це зробити з алфавітно-цифрового термінала або його емулятора. Щоб перейти з графічної оболонки в алфавітно-цифровий термінал, натисніть ***<Ctrl> + <Alt> + <Fx>***, де ***Fx*** – одна з функціональних клавіш ***F1, F2 ... F6***. Ви потрапите до буквено-цифрової консолі. Номер консолі вказує покажчик ***tty***. Наприклад: ***tty2 –*** означає, що ви перебуваєте в другій консолі. За замовчуванням таких консолей 7. Причому сьома консоль – графічна. Переходити між консолями можна натискаючи клавіші ***<Alt> + <Fx>***.

Потренуйтеся переходити між консолями

У кожній консолі вам пропонують реєструватися. Таким чином у різних консолях ви можете працювати під різними іменами і, що ще більш важливо, з різними правами. Отже, реєструємося в консолі.

На екрані з'являється напис, що пропонує ввести реєстраційне ім'я, як правило, це ***„ login: ”***. Набравши своє реєстраційне ім'я, натисніть клавішу ***<Enter>***. Система запросить у вас пароль, який відповідає введеному імені, видавши спеціальне запрошення, зазвичай ***„ Password: ”***. Уважно наберіть пароль, встановлений для вас системним адміністратором, і натисніть клавішу ***<Enter>.*** Пароль на екрані не відображається, тому набирайте його акуратно! Якщо все було зроблено правильно, у вас на екрані з'явиться запрошення до введення команд операційної системи.

Команда man – універсальний довідник

По ходу вивчення операційної системи Linux вам часто буде необхідна інформація про те, що робить та чи інша команда або системний виклик, які у них параметри і опції, для чого призначені деякі системні файли, який їхній формат і т.д.

Для отримання довідки служить утиліта ***man***

Користуватися утилітою man досить просто наберіть команду:

man *ім'я* ,

де ***ім'я*** – це ім'я команди, утиліти, системного виклику, бібліотечної функції або файлу, що вас цікавить.

Спробуйте за допомогою цієї команди подивитися інформацію про команди: ***pwd, who, whoami, last***.

Щоб перегорнути сторінку отриманого опису, якщо воно не влізло на екрані повністю, слід натиснути клавішу ***<пробіл>***. Для прокрутки одного рядка скористайтеся клавішею ***<Enter>***. Повернутися на сторінку назад дозволить одночасне натискання клавіш ***<Ctrl>*** і ***<b>***. Вийти з режиму перегляду інформації можна за допомогою клавіші ***<q>***.

Іноді імена команд інтерпретатора і системних викликів збігаються. Тоді щоб знайти цікаву для вас інформацію, необхідно вказати для утиліти ***man*** категорію, до якої належить ця інформація (номер розділу). У Linux прийнято наступний поділ:

1. Виконувані програми або команди оболонки (shell).
2. Системні виклики (функції, що надаються ядром).
3. Бібліотечні виклики (функції, що надаються програмними бібліотеками).
4. Спеціальні файли (зазвичай знаходяться в каталозі */dev*).
5. Формат системних файлів і прийняті угоди.
6. Ігри (зазвичай відсутні).
7. Різне (включає пакети макросів і угоди), наприклад man (7), groff (7).
8. Команди адміністрування системи (зазвичай запускаються тільки суперкористувачем).
9. Процедури ядра (нестандартний розділ).

Якщо ви знаєте розділ, до якого відноситься інформація, то утиліту man можна викликати в Linux з додатковим параметром:

man номер-розділу ім'я

В інших операційних системах цей виклик може виглядати інакше. Для отримання точної інформації про розбиття на

розділи, форму вказівки номера розділу і додаткових можливостей утиліти man наберіть команду:

man man

Використовуючи команди ***who, whoami, last*** отримаєте відомості про користувачів, які працюють у системі.

Завершення роботи в Linux

В ОС Linux не можна виключати комп'ютер простим відключенням живлення. Справа в тому, що в будь-який момент часу в системі запущено кілька процесів. Деякі з цих процесів можуть працювати з файлами, причому система не записує всі зміни файлів на диск відразу після внесення цих змін користувачем або процесом, а зберігає їх тимчасово в оперативній пам'яті (кешує). Якщо просто вимкнути живлення, ці зміни не будуть збережені і пропадуть, що іноді може призвести навіть до неможливості подальшого завантаження системи. Так що треба вміти правильно завершити роботу системи перед вимиканням комп'ютера. Це робиться командою ***shutdown***.

Команда ***shutdown*** може бути виконана тільки користувачем ***root***.

З опцій програми ***shutdown*** найбільш часто використовують дві:

-h – повна зупинка системи (комп'ютер буде вимкнений);

-r – перезавантажити систему.

Коли ви захочете просто вимкнути комп'ютер. Еквівалентом команди ***shutdown -h*** є команда ***poweroff***. Еквівалентом команди ***shutdown -r*** є команда ***reboot***.

Завдання до виконання

1. Спробуйте за допомогою команди ***man*** подивитися інформацію про команди: ***who, whoami, last, date, cal, tty, ls, clear, ps, mc***.
2. Використовуючи команди ***who, whoami, last,*** отримайте відомості про користувачів, які працюють у системі.
3. Виконайте перезавантаження системи.
4. Виконайте завершення роботи.

Контрольні питання

1. Перерахуйте основні функції та призначення багатозадачної та багатокористувацької операційної системи LINUX і її відмінні особливості від однопрограмної системи DOS.
2. Яке призначення має ядро системи та інтерпретатор команд?
3. У чому полягає поняття «процес» і які операції можна виконати над процесами?
4. Як задаються і виконуються прості і складні команди?
5. Які функції виконує командний інтерпретатор *Shell*?

Робота № 2. Файлова система Unix. Основні операції над файлами

Мета роботи: оволодіння практичними навичками роботи в системі Linux. Знайомство із структурою файлової системи, основними командами роботи з файлами.

Короткі теоретичні відомості

Файлова система є елементом для організації зберігання даних, необхідних для функціонування ОС, а також даних прикладних програм і користувачів. Зазвичай всі дані зберігаються в логічно розділених порціях – файлах, які по суті є міткою, яка присвоюється порції користувачем або процесом. Директорії є спеціальними файлами, які точно також об'єднують призначенням міткою сукупність файлів, це і є ім'ям директорії. Організація файлів і директорій виконується у вигляді ієрархічної структури, початком якої є коренева директорія /, від якої беруть початок всі інші директорії. В операційних системах Windows коренева директорія також існує, але є прихованою і від неї безпосередньо беруть початок логічні диски. Робота з ієрархією дозволяє розширювати логічну структуру файлової системи підключенням інших файлових систем у точках монтування, не змінюючи поточну організацію файлової системи. Процес роботи з даними в файловій системі організовується за допомогою системних викликів, які передаються в ядро для виконання і взаємодіють безпосередньо із структурами файлової системи для знаходження необхідних блоків даних, а потім проводять операції на пристрої зберігання. Для прикладних процесів взаємодія складається з високорівневих операцій: відкриття файлу, позиціонування в файлі, читання даних з файлу, запис у файл, закриття файлу, створення директорії, видалення директорії і т. п.

Операційна система Linux підтримує широкий спектр файлових систем. Кожна з них має свої переваги й недоліки, які визначають їх використання в кожному конкретному випадку. Спосіб управління файловими системами однаковий незалежно від її типу.

До числа підтримуваних файлових систем в операційних системах GNU Linux відносяться:

- Ext2
- Ext3
- Ext4
- Xfs
- Btrfs
- Zfs
- Reiserfs
- Fat32

Ext2 – файлова система, вперше представлена в 1992 році, основними особливостями є використання індексних дескрипторів для роботи з файлами, включаючи зберігання метаінформації і адресації блоків даних. Блоки об'єднані в групи блоків для забезпечення близькості зберігання пов'язаних даних.

Ext3 – файлова система, яка усунула основний недолік ext2 – відсутність можливості відновлення після збоїв. Ext3 доповнена можливістю ведення журналів операцій, які можуть бути задіяні при збої, що значно підвищило надійність зберігання даних. Також були розширені ліміти зберігання структур файлової системи у порівнянні з ext2.

Ext4 – наступна версія файлової системи ext3. Основні поліпшення спрямовані на підвищення продуктивності роботи. Такі як введення екстентів – послідовних записаних блоків даних для ефективного управління великими файлами, в цьому випадку немає необхідності зберігання адресної інформації для кожного блоку, тільки початок екстента і його довжину в блоках. До інших поліпшень належать ще більше збільшення лімітів для структур файлової системи, наприклад, максимальний розмір файлу складає 16 Тб., зняття обмеження на кількість підкаталогів (в ext3 32000). Відкладене розподілення блоків дозволяє підвищити продуктивність шляхом послідовного запису великого числа кешованих даних, зменшуючи тим самим фрагментацію даних.

XFS – журнальована файлова система, розроблена спеціально для ОС IRIX, пізніше була опублікована під ліцензією

GNU GPL, внаслідок чого отримала поширення серед Linux дистрибутивів. Спочатку XFS проєктувалася для використання в дисках великого розміру, тому використовується 64 бітна адресація. Принципи роботи схожі з уже описаною ext4, до основних з яких відноситься використання індексних дескрипторів, облік об'єму екстентами, широке використання бінарних дерев для зберігання структур файлової системи. Як основний недолік експерти виділяють занадто активне використання кешування, що може призводити до втрат великої кількості активних, але не закінчених операції при збоях електроживлення.

Btrfs – файлова система розробляється компанією Oracle. Принцип роботи полягає в активному використанні B-дерев і механізму копіювання при записі. Має низку переваг у порівнянні з уже розглянутими файловими системами. Серед них виділяється підтримка на рівні файлової системи програмної реалізації RAID, створення моментальних знімків, можливості по клонуванню, стиснення даних.

Zfs – файлова система, яка розроблена і використовується в операційній системі Solaris. Є проєктом з відкритим вихідним кодом. До основних особливостей можна віднести сумістність файлової системи і менеджера томів. Є 128 бітною файловою системою, що дає можливість значно розширити ліміти внутрішніх структур, таких як: розмір файлової системи, кількість файлів, розмір одного файлу, розмір атрибута і т. п. При роботі використовуються пули зберігання, які дозволяють об'єднувати кілька фізичних пристроїв в один том і вже на його основі створювати файлову систему. Zfs підтримує свій власних варіант побудови відмовостійкості сховища на основі технології RAIDZ з можливістю гарячого резерву. Є можливість виділення окремих фізичних пристроїв для виконання кешування, що актуально в даний час у зв'язку з широким розвитком твердотільних накопичувачів. Значне поширення в ZFS має технологія створення моментальних знімків і клонів файлової системи. Завдяки використанню моделі транзакцій на основі копіювання при записі,

знімки і клони створюються практично миттєво і ними зручно користуватися для цілей резервного копіювання та управління файловими системами. Наскрізний контроль цілісності даних дозволяє визначати коректність даних кожен раз при їх зчитуванні. Використання механізму дедуплікації дозволяє на рівні файлової системи визначати дублюючі дані і зберігати тільки один їх примірник, зберігаючи тим самим дисковий простір.

ReiserFS **–** перша журнальована файлова система, підтримка якої включена в ядро ОС Linux. Основними особливостями є можливість зміни розміру файлової системи без необхідності її розмонтування. Володіє схильністю до фрагментації, при цьому існують способи дефрагментації, але вони вкрай неефективні. В даний момент вважається застарілою файловою системою навіть за умови розробки нової версії, що має назву Reiser4. У той же час має досить високу продуктивність операцій над файлами малого розміру.

Fat32 – файлова система, що має значне поширення в середовищі Windows. Є подальшим розвитком файлової системи Fat, яку вдосконалили для збільшення розміру тому шляхом зміни розрядності адресації блоків даних (у термінах Fat-кластерів) з 16 до 32. При цьому всі інші принципи роботи залишилися незмінними для забезпечення сумісності. Розділ при форматуванні його під файлову систему fat32 поділяється на дві області: системну і простір для зберігання даних. Системна область містить структури, необхідні для управління розміщенням даних файлів і каталогів. Також тут міститься кореневий каталог і завантажувальний сектор для завантаження ОС. Кластера файлу містяться в просторі для зберігання даних. Інформація про зайнятих файлом кластерах міститься в таблиці розміщення файлів (File Allocation Table – FAT) завдяки цій структурі файлова система і отримала свою назву. Кожному кластеру простору даних відповідає запис в FAT. Для адресації всього набору кластерів файлу використовується ланцюжок, в якому в запису відповідному зайнятому кластеру міститься адреса наступного кластера файлу, а в разі

вільного кластера міститься нульове значення. Таблиця FAT міститься в двох примірниках внаслідок особливого значення для роботи файлової системи, з огляду на інтенсивне використання вона також завантажується в оперативну пам'ять і зберігається там максимально довго. Основним недоліком є досить повільна роботи з файлами через виділення першого вільного кластера, що призводить до створення складних ланцюжків кластерів і фрагментації. Також відзначається низька відмовостійкість, оскільки журнал файлова система не веде.

NTFS – сучасна файлова система, поширена в середовищі Windows. Специфікація файлової системи є закритою, що ускладнило створення програмного коду для доступа інших ОС. Проте методом зворотної розробки фахівцями написані драйвера для доступа з інших операційних систем. Прийшла на зміну файлової системи Fat32. Має наступні поліпшення: зберігання метаданих, журнал роботи, списки контролю доступу, квотування. Основною структурою для зберігання інформації про файли є MFT (Master File Table). Під зберігання цієї таблиці виділяється ціла область для запобігання її фрагментарності в процесі роботи файлової системи. Кожен запис MFT має фіксований розмір і включає в себе інформацію у вигляді імені файлу, розміру, набору атрибутів, а також розміщення на диску окремих фрагментів. Якщо для зберігання інформації про файл недостатньо одного запису MFT, то може бути задіяно кілька записів, які можуть розташовуватися непослідовно. Атрибути можуть бути резидентними, тобто зберігатися повністю в запису MFT, або нерезидентними, які зберігаються за межами області MFT. Для адресації кластерів можливе використання 64 розрядних ідентифікаторів, але в поточних версіях використовуються тільки 32.

Короткий опис основних команд для роботи з файлами

Тут наведено команди не в повному обсязі, більш повний опис дивіться в довідкових посібниках (man <команда>).

ls [-alrstu] [ім'я] – виведення вмісту каталогу на екран. Якщо параметр не вказано, видається вміст поточного каталогу.

Аргументи команди:

-a – виводить список усіх файлів і каталогів, у тому числі і прихованих;

-l – виводить список файлів у розширеному форматі, показуючи тип кожного елемента, повноваження, власника, розмір і дату останньої модифікації;

-r – виводить список у порядку, зворотному заданому;

-s – виводить розміри кожного файлу;

-t – перераховує файли і каталоги відповідно до дати їх останньої модифікації;

-u – перераховує файли і каталоги в порядку, зворотному до них по останній модифікації.

cat <ім'я файлу> – виведення вмісту файлу на екран. Команда ***cat > text.1*** створює новий файл з ім'ям ***text.1***, який можна заповнити символьними рядками, вводячи їх з клавіатури. Натискання клавіші ***Enter*** створює новий рядок. Завершення введення – натискання ***Ctrl - d***. Команда ***cat text.1> text.2*** пересилає вміст файлу ***text.1*** в файл ***text.2***. Злиття файлів здійснюється командою ***cat text.1 text.2 > text.3***

rm <ім'я файлу> – видалення файлу (файлів). Команда ***rm text.1 text.2 text.3*** видаляє файли ***text.1, text.2, text.3***. Інші варіанти цієї команди – ***rm text.[123]*** або ***rm text.nc[1-3]***.

wc <ім'я файлу> – виведення числа рядків, слів та символів у файлі.

clear – очищення екрану.

cd [каталог] – змінює поточний каталог на зазначений. Якщо параметр опущений, то поточним стає домашній каталог.

mkdir <каталог> – створює каталог.

rmdir <каталог> – видаляє каталог.

cp [-Rp] файл1 файл2

cp [-Rp] файл ... каталог – копіює один файл в інший (затираючи колишній зміст цього іншого файлу) або копіює файли в зазначений каталог. Ключ ***-R*** призначений для копіювання каталогів, ключ ***-p*** дозволяє зберігати власників файлів, режим доступу та час доступу і зміни.

mv файл1 файл2

mv файл ... каталог – перейменовує файл або переміщує файли в заданий каталог.

Групування команд

Групи команд або складні команди можуть формуватися за допомогою спеціальних символів (метасимволів):

& – процес виконується у фоновому режимі, не чекаючи закінчення попередніх процесів;

? – шаблон, поширюється тільки на один символ;

***** – шаблон, поширюється на всі подальші символи;

| – програмний канал – стандартний вихід одного процесу є стандартним входом іншого;

> – переадресація виведення в файл;

< – переадресація введення з файлу;

; – якщо в списку команд команди відокремлюються одна від одної точкою з комою, то вони виконуються одна за одною;

&& – ця конструкція між командами означає, що наступна команда виконується тільки при нормальному завершенні попередньої команди (код повернення 0);

|| – подальша команда виконується тільки якщо не виконалась попередня команда (код повернення 1);

() – групування команд у дужки;

{} – групування команд з об'єднаним висновком;

[] – вказівка діапазону або явне перерахування (без ком);

>> – додавання вмісту файлу в кінець іншого файлу.

Завдання до виконання

1. Ознайомтесь з документацією, звернувши увагу на такі питання:

- команди входу в систему, зміни пароля, одержання системної підказки, виводу календаря і зміни дати;
- організацію і структуру файлової системи Linux, обмеження на імена файлів;
- типи файлів, каталоги і посилання;
- системні каталоги;
- створення, видалення, копіювання і перегляд вмісту файлів.

2. Ознайомтесь з такими командами UNIX-подібних систем (рекомендується застосовувати сторінки man): ***man, passwd, date, cat, more, wc, who, ls, cd, cal, cp, mv, mkdir, rm, rmdir***

3. У Linux перейдіть у текстову консоль і зареєструйтесь там.

4. Створіть каталог, назвавши його вашим прізвищем.

5. Усередині цього каталогу створіть два підкаталоги – KAT1 і KAT2.

6. Відкрийте каталог KAT1 і створіть в ньому файл F1 з текстом.

7. Створіть файл F2 з текстом.

8. Приєднайте файл F1 до файлу F2.

9. Створіть новий файл F3, скопіювавши в нього файл F2.

10. Скопіюйте файли F1 і F2 в каталог KAT2.

11. Перевірте правильність виконання операції.

12. Видаліть всі файли з каталогу KAT1.

13. Видаліть каталог KAT1.

14. Перейменуйте файл F1 в файл F1.txt.

Контрольні питання

1. Що вважається файлами в ОС LINUX?

2. Що визначає атрибути файлів і яким чином їх можна переглянути та змінити?

3. Які методи створення і видалення файлів, каталогів ви знаєте?

4. У чому полягає пошук за шаблоном?

5. Якою командою можна отримати список працюючих користувачів і зберегти його у файлі?

Робота № 3. Перенаправлення вводу/виводу. Конвеєри команд. Редактор sed. Утиліти cat, head, tail, sort

Мета роботи: оволодіння практичними навичками перенаправлення стандартних потоків, роботи з фільтрами й організації конвеєрів.

Короткі теоретичні відомості

Найважливішим із призначених для користувача процесів є командна оболонка (вона ж командний інтерпретатор, або просто shell). Саме вона забезпечує взаємодію користувача з системою в текстовому режимі, дозволяючи вводити команди. Саме вона запускається, коли ви входите на текстовій консолі, і надає вам інтерфейс командного рядка.

Не потрібно, захопившись зручностями графічного інтерфейсу, недооцінювати командний рядок. По-перше, багато адміністративних завдання можуть бути виконані тільки звідти; по-друге, командний рядок – найзручніший засіб автоматизації рутинних процедур.

Командою в Linux вважається все, що може бути виконано: виконувані файли, вбудовані команди оболонки, псевдоніми команд, призначені для користувача функції, файли сценаріїв (скрипти) – заздалегідь підготовлені послідовності команд у текстовому вигляді.

Оболонка приймає від користувача команди, обробляє, якщо потрібно, їх аргументи, відправляє команди на виконання, приймає повернені ними значення і виконує певні дії в залежності від цих значень. Крім того, в оболонку вбудована мова програмування (командна мова), що дозволяє писати складні розгалужені командні сценарії. Саме командна мова відрізняє різні оболонки одну від другої. Список всіх встановлених у системі програм-оболонок знаходиться в файлі */* ***etc/shells***.

Ще одна можливість оболонок – перенаправлення потоків введення-виведення. Як правило, більшість команд (утиліт) приймає інформацію з клавіатури або з файлу, якщо його

вказано як параметр, і виводить результати на екран. Однак фактично вони працюють із так званими стандартними потоками введення і виведення, які пов'язані з певними файлами. Файл в UNIX-подібних системах розглядається як потік байт. Оскільки пристрої в UNIX-подібних системах розглядаються як файли, а операції введення і виведення – як зчитування і записування у відповідні файли, це дозволяє легко переводити вхідний і вихідний потоки з файлів на пристрої чи навпаки.

Стандартний потік введення за умовчанням зчитується з клавіатури. Якщо як параметр указано ім'я файлу, то замість стандартного введення відповідна утиліта буде організовувати вхідний потік з указаного файлу (але так діють не всі команди!) Вихідних потоків є два – стандартний потік виведення (за умовчанням у сучасних системах – на екран монітору), і потік повідомлень про помилки (за умовчанням – туди ж).

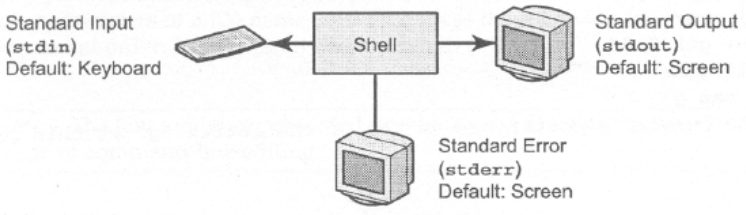


Рис.1 Стандартні потоки введення-виведення

Всі потоки можна перенаправити в інший файл. Це може бути файл на диску, файл пристрою (наприклад, принтер або /dev/null) або стандартний потік іншого процесу.

Символ **'<'** перенаправляє вхідний потік. Після цього символу очікується ім'я файлу або пристрою, з якого буде братись вхідний потік.

Наприклад, команда ***cat*** без параметрів очікує введення з клавіатури і кожний рядок передає на екран монітора. Команда ***cat my_file*** замість введення з клавіатури виведе на екран вміст файлу ***my_file***, якщо такий існує, і повідомлення про помилку, якщо такого не існує.

Команда ***cat < my_file*** на перший погляд буде робити те ж саме, але з точки зору операційної системи і самої утиліти ***cat*** все буде відбуватись по-іншому. В попередньому випадку командна оболонка запускала утиліту ***cat*** і передавала їй параметр командного рядка ***my_file***, а сама утиліта ***cat*** вже інтерпретувала цей параметр як ім'я вхідного файлу. В останньому випадку командна оболонка запускала утиліту ***cat*** без параметрів, але передавала їй вміст файлу ***my_file*** в якості вхідного потоку.

Команда ***cat > my_file*** перенаправляє вихідний потік. Оскільки вхідний потік не перенаправляється, введення буде очікуватись з клавіатури. Тому ця команда створить файл ***my_file***, якщо він не існує, знищить вміст файлу ***my_file***, якщо він існує, і буде записувати в цей файл усе, що буде введено з клавіатури, аж поки не поступить символ кінця файлу EOF (Ctrl+D). Існує також можливість дописати інформацію в кінець файлу, не знищуючи його вмісту. Така команда буде мати вигляд ***cat >> my_file***

Потік повідомлень про помилки в оболонці *sh* (і *bash*) перенаправляється окремо, він позначається **2>**. Наприклад, команда

cat < my_file1 > my_file2 2> my_file3

у разі, якщо файл ***my_file1*** існує, запише його вміст в файл ***my_file2***, а якщо не існує, то запише повідомлення про помилку в файл ***my_file3***.

Дуже часто потік помилок намагаються взагалі «загасити». Для того, щоб знищити якийсь потік, існує спеціальний пристрій ***/dev/null***. Все, що в нього направляється, зникає безслідно.

Перенаправления введення-виведення широко використовується у двох випадках. Перший – це запуск утиліт у фоновому режимі. Другий – це використання спеціальних команд-утиліт, які призначені саме для того, щоб прийняти певну інформацію з одного файлу, обробити її, а результат записати у другий файл. Такі утиліти називаються фільтрами. Утиліта ***cat***, варіанти використання якої з перенаправлени-

ям потоків було розглянуто вище – це простіший фільтр. Він практично не обробляє інформацію, лише може зчіплювати кілька файлів в один. Інші корисні фільтри: ***cut, grep, sort***.

Утиліта ***cut*** переглядає вхідний файл, і виділяє з кожного його рядка інформацію за ознаками розміщення в певних колонках або полях. Наприклад, рядки файлу ***/etc/passwd*** розділяються на поля за допомогою символу "**:**" Перше поле – *login*, п'яте поле – інформація про користувача. Якщо ми хочемо надрукувати лише цю інформацію, ми можемо дати команду:

cut -d: -f1,5 < /etc/passwd

Ключ ***-d*** задає символ-роздільник полів (у цьому випадку "**:**"), ключ ***-f*** – список полів, що треба роздрукувати (у цьому випадку 1 і 5).

Утиліта ***head [-n count] [file ...]*** виводить перші ***count*** рядків файлу (за замовчуванням 10).

Утиліта ***tail [-f] [-n count] [file ...]*** виводить останні ***count*** рядків файлу (за замовчуванням 10). Якщо зазначений ключ ***–f***, то очікує додавання даних в кінець файлу і виводить їх.

Утиліта ***sort [-c | -m] [-o output] [-urnb] [file ...]*** виконує сортування рядків файлів, їх об'єднання або перевіряє відсортований файл чи ні. Значення параметрів:

-c – тільки перевірити правильність сортування

-m – об'єднати попередньо відсортовані файли

-u – видаляти повторювані елементи

-r – сортування в зворотному порядку

-n – сортування чисел

-b – ігнорувати лідируючі пропуски

-o file – виконати виведення в файл *file*

Утиліта ***wc [-c | -m] [- lw] [file ...]*** читає один або більше вхідних файлів і, за замовчуванням, виводить число символів, слів і байт, що містяться в кожному файлі на стандартний вихід. Значення параметрів:

-c – вивести число байт у кожному вхідному файлі

-l – вивести число символів нового рядка в кожному вхідному файлі

-m – вивести число символів у кожному вхідному файлі

-w – вивести число слів у кожному вхідному файлі.

Утиліта ***grep*** виводить лише ті рядки, в яких зустрічається заданий рядок пошуку. Утиліта ***uniq*** – видаляє повторювані рядки з відсортованого файлу, а утиліта ***tr*** – замінює одні символи на інші. Докладніше про ці та інші фільтри вам слід дізнатися з довідкової системи ***man***.

Існує можливість перенаправити вихідний потік однієї утиліти безпосередньо у вхідний потік іншої, без використання тимчасових файлів. Це так звані конвеєри (ріре). В UNIX-подібних системах усі утиліти, що поєднані в конвеєр, запускаються паралельно і обробляють інформацію по мірі її надходження. Конвеєр утворюється за допомогою символу ‘ | ’ таким чином:

util1 | util2 | util3

При утворенні конвеєра окремо перенаправляти вхідні й вихідні потоки на проміжних стадіях не можна – це буде або сприйнято як синтаксична помилка, або результат може бути непередбачуваним.

Приклад конвеєру:

ps –a1 | grep root | more

Команда ***ps*** з ключами ***–a1*** направить у вихідний потік список всіх процесів у системі, ***grep root*** вибере з них лише ті, які виконуються від імені ***root***, ***more*** забезпечить їх виведення на екран посторінково.

Інший приклад:

cat /etc/passwd | cut -d: -f1,5 | more

Ця команда зробить те ж саме, що й приклад з командою cut, який розглядався раніше, але виведення на екран буде посторінковим.

Якщо проміжні результати на деякій із стадій конвеєра бажано зберегти, можна скористатись командою ***tee my_file***. Ця команда візьме вхідний потік, передасть його без змін у вихідний потік і одночасно продублює у файл *my_file*. Наприклад, так можна модифікувати один із розглянутих вище прикладів:

ps -ef | grep root | tee my_file | more

Тепер ми не лише побачимо на екрані посторінково виведений список всіх процесів ***root***, але й збережемо його у файлі ***my_file***.

Завдання до виконання

1. Ознайомтесь з документацією, звернувши увагу на такі питання:

- Командні оболонки, їх запуск, конфігураційні файли.
- Стандартні потоки та їх перепризначення.
- Організація конвеєрів.
- Організація фільтрів і команди, використовувані як фільтри.

2. Ознайомтесь з такими командами UNIX: ***tee, find, cut, date, grep, sort.***

3. Зверніть увагу на метасимволи ****, ?, /, […], $*** і на правила інтерпретації їх при використанні одинарних ***'…'*** та подвійних лапок ***«…»***.

4. Ознайомтесь з використанням в командах операторів перепризначення потоків *«>»*, *«<»* і організації конвеєрів *«|»* та використанням псевдопристрою ***/dev/null***.

Контрольні питання

1. Для яких цілей використовується перепризначення потоків?
2. В яких випадках застосовується конвеєрна обробка?
3. Для чого застосовуються параметри командного рядка?
4. Яка інформація про процеси виводиться командою *ps*?

Робота № 4. Характеристики та особливості X Window. Віконні менеджери, середовища KDE та GNOME

***Мета роботи**:* отримання основних навичок по роботі та налаштуванню графічних віконних менеджерів X Window, KDE, GNOME.

Короткі теоретичні відомості

Для виконання деяких завдань на комп'ютері не обійтися без використання зручного графічного інтерфейсу. Такими завданнями можуть бути робота з графічними редакторами, відеоредакторами, запуск ігрових додатків, візуалізація, анімація тощо. Для всього цього необхідно програмне забезпечення (ПЗ), ресурси (апаратні, програмні), специфікації (протоколи взаємодії), які реалізують користувачам повноцінний графічний інтерфейс (GUI).

Оскільки ОС UNIX / Linux мають архітектуру, ключовою особливістю якої є мережева клієнт-серверна взаємодія, то реалізація графічної системи також побудована на схемі «клієнт-сервер». Ядром такої системи є протокол, що описує правила організації і надання певним сервером X своїх потужностей і обчислювальних ресурсів клієнтам – машинам користувачів, на яких запускається графічна оболонка, що дозволяє працювати додаткам у вигляді вікон (Window) з характерними для них органами управління та оформленням. Причому зовнішній вигляд і супутні функціональні можливості GUI будуть залежати від відповідних графічних бібліотек, установлених на клієнтській машині. Все це в сукупності визначає графічну систему X Window.

X Window забезпечує роботу графічного оточення завдяки всього трьом основним її компонентам: X-серверу, диспетчеру дисплеїв, менеджеру вікон. Диспетчер дисплеїв авторизує і / або реєструє користувачів для надання їм середовища для запуску (набір сценаріїв запуску) системи. Також диспетчер дисплеїв управляє роботою X-сервера, тобто він може запускати його коли клієнт відправив відповідний запит. У свою чергу X-сервер по завершенні запуску надає клієнтам абстракт-

ний інтерфейс для пристроїв введення (миша, клавіатура), а також для відтворення растрових зображень. Менеджер вікон служить для організації додатків у вигляді вікон, дозволяючи перемикатися між ними, згортати / розгортати, відкривати, переміщувати по екрану, управляти робочими столами. Таким чином, виступаючи в ролі інтерактивного візуалізатора середовища користувача і додатків. Зовнішній вигляд додатків реалізується за допомогою різних бібліотек інтерфейсних елементів (наприклад GTK +), які дозволяють перетворювати вікна, меню, заголовки, кнопки та інші органи управління.

Гнучкість, масштабованість, а також відкритість системи X Window породила безліч реалізацій не тільки її самої (і її протоколу), але також не менше число графічних середовищ. Довгий час (та й донині) оригінальна графічна оболонка на «чистому X11» з власною бібліотекою інтерфейсних елементів була незавершеною, так як з самого початку не малося на увазі її повсюдне використання. Однак, з іншого боку клієнт-серверна архітектура X Window, а також її незалежність від низькорівневої реалізації GUI на стороні клієнта «розв'язували» руки розробникам ПЗ, які дуже часто прагнули забезпечити свої додатки оригінальним зовнішнім виглядом, розробляючи і поставляючи разом з ПЗ також власні бібліотеки елементів GUI. При цьому розвиток як самої системи X Window, так графічних середовищ і бібліотек елементів інтерфейсу відбувається незалежно один від одного зусиллями різних розробників.

Саме тому зараз можна спостерігати таку кількість всіляких графічних оточень для UNIX / Linux. Деякі з них (KDE, GNOME, XFCE) за якістю дизайну, зручністю використання, функціоналу і швидкодії досить зручні, що люди, що вперше побачили і попрацювали в цих середовищах, встановлюють і використовують їх як основні.

KDE – потужна графічна оболонка, що включає в себе набір прикладних програм, достатній для задоволення потреб звичайного користувача. Є навіть повнофункціональний офісний пакет KOffice, який за своїми можливостями

не поступається більш знаменитому OpenOffice.org. Ця оболонка входить в стандартну поставку дистрибутивів Kubuntu, OpenSuSE, ASPLinux, ALTLinux, MOPS, Mandriva, Debian.

KDE надзвичайно популярний завдяки різноманітності програмного забезпечення, що входить до його складу. Так, набір додатків дистрибутива MOPS фактично складається тільки з ПО, вбудованого в KDE, і при цьому отримується система досить функціональна, щоб задовольнити потреби більшості користувачів.

Крім офісного пакету в KDE входять браузер, поштовий клієнт і навіть програма для роботи в мережі BitTorrent. Прагнення творців цієї оболонки зробити самодостатнє робоче середовище призвело до того, що фірмовий центр управління дозволяє визначати ряд загальносистемних параметрів. А конфігурації шрифтів і розкладок, які використовуються цим інтерфейсом, мають пріоритет над звичайним способом, що задається для Linux.

З іншого боку, програми KDE практично не знайомі користувачам Windows, оскільки не є кросплатформеними. Через це можуть виникнути певні складнощі при переході між ними.

Центр управління – особлива гордість KDE. З його допомогою можна налаштувати все, що в принципі налаштовується: від мережевих інтерфейсів до тла робочого столу. Його концепція особливо приваблива для початківців, оскільки всі необхідні інструменти зібрані в одній оболонці.

Однак ряд сучасних дистрибутивів (Mandriva, OpenSuSE, ALTLinux, Linux XP) мають свої центри настройки, що знижує споживчу цінність аналогічної програми, що входить до складу KDE. Фактично вона дублює основні функції штатного налагоджувальника.

Головним недоліком KDE прийнято вважати високі вимоги до апаратної конфігурації. Дійсно, на старих і малопотужних машинах цей інтерфейс краще не використовувати.

GMOME – давній конкурент KDE. Суперечки між прихильниками і противниками використання цих робочих середовищ вже переросли в чергову Holy War. Йдуть вони і до цього дня, тому можна сміливо сказати, що вирішаль-

ної переваги немає ні в одного інтерфейсу. GMOME використовується в дистрибутивах Ubuntu, Linux XP, OpenSuSE, ASPLinux, Fedora, Mandriva, Debian.

Головна перевага цього робочого середовища полягає в тому, що воно побудовано на бібліотеці GTK. Цей інструментарій поширюється по гнучкій ліцензії, що допускає його використання комерційними програмами. Тому теоретично потенціал GNOME досить високий. Якщо Linux займе солідну нішу в корпоративному секторі, то швидше за все ряд розробників затребуваних прикладних програм не зможуть взяти GPL і будуть орієнтуватися на моделі, що дозволяють отримувати серйозний прибуток. Треба сказати, що саме GTK застосовувалася при створенні досить популярних програм Firefox і Thunderbird.

Налаштовується GNOME за допомогою централізованої системи GConf. Вимоги у GNOME і KDE до ресурсів машини приблизно однакові. Комп'ютер, на якому працює одна оболонка, придатний і для використання другої.

Завдання до виконання

1. Вивчіть основні елементи KDE (Іконки, кнопки швидкого запуску додатків, перемикачі віртуальних робочих столів, панель задач, К Меню).
2. Запишіть основні назви елементів KDE.
3. Запишіть назви кнопок швидкого запуску додатків. Які програми вони запускають?
4. Відкрийте KWord і наберіть наступний текст: «The Quick Brown Fox Jumps Over The Lazy Dog», використовуючи два різні стилі за вашим вибором. Збережіть цей файл в домашньому каталозі користувача, закрийте KWord.
5. Отримайте довідку про об'єкт, що вас цікавить.
6. Створіть будь-який малюнок з допомогою Paint, щоб в ньому були всі фігури (1. еліпс, 2. окружність, 3. лінія, 4. прямокутник, 5. коло) хоча б по одному разу і були присутні не менше чотирьох кольорів. Збережіть файл з малюнком в домашньому каталозі.

Контрольні питання

1. Перерахуйте стандартні функції KDE.
2. Що є компонентом робочого столу KDE?
3. Назвіть функції панелі робочого столу.
4. Як отримати довідку в діалоговому режимі?
5. Які функції надає центр керування KDE?

Робота № 5. Розробка скриптів на bash. Змінні інтерпретатора, умовні та циклічні конструкції

Мета роботи: оволодіння практичними навичками професійної роботи з командною оболонкою shell – використання змінних і створення командних файлів.

Короткі теоретичні відомості

У попередніх роботах взаємодія з командним інтерпретатором Shell здійснювалося за допомогою командного рядка. Однак Shell є також і мовою програмування, який застосовується для написання командних файлів (shell-файлів). Командні файли також називаються скриптами і сценаріями. Shell-файл містить одну або кілька виконуваних команд (процедур), а ім'я файлу в цьому випадку використовується як ім'я команди.

Змінні командного інтерпретатора

Для позначення змінних Shell використовується послідовність літер, цифр і символів підкреслення; змінні не можуть починатися з цифри. Присвоєння значень змінним проводиться з використанням знаку =, наприклад, PS2 = '<'. Для звернення до значення змінної перед її ім'ям ставиться знак $. Їх можна розділити на наступні групи:

– позиційні змінні виду *$n*, де *n* – ціле число;

– прості змінні, значення яких може задавати користувач або вони можуть встановлюватися інтерпретатором;

– спеціальні змінні #, ?, -, !, $ встановлюються інтерпретатором і дозволяють отримати інформацію про кількість позиційних змінних, код завершення останньої команди, ідентифікаційний номер поточного і фонового процесів, а також про поточні прапорці інтерпретатора Shell.

Прості змінні. Shell присвоює значення змінним:

```
z = 1000
x = $z
echo $x
1000
```

Тут змінній *x* присвоєно значення *z*.

Позиційні змінні. Змінні виду $*n*, де *n* – ціле число, використовуються для ідентифікації позицій елементів у командному рядку за допомогою номерів, починаючи з нуля. Наприклад, у командному рядку

cat text_1 text_2 ... text_9

аргументи ідентифікуються параметрами *$1...$9*. Для імені команди завжди використовується *$0*. В даному випадку *$0* – це *cat*, *$1 – text_1*, *$2 – text_2* і т. д. Для присвоювання значень позиційним змінним використовується команда *set*, наприклад:

set arg_1 arg_2 ... arg_9

тут *$1* присвоюється значення аргументу *arg_1*, *$2 – arg_2* і т. д.

Для доступу до аргументів використовується команда *echo*, наприклад:

echo $1 $2 $9

arg_l arg_2 arg_9

Для отримання інформації про всі аргументи (включаючи останній) використовують метасимвол *. Наприклад:

echo $*

arg_2 arg_3 ... arg_10 arg_11 arg_12

За допомогою позиційних змінних Shell можна зберегти ім'я команди та її аргументи. При виконанні команди інтерпретатор Shell повинен передати їй аргументи, порядок яких може регулюватися також за допомогою позиційних змінних.

Спеціальні змінні. Змінні *-, ?, #, $, !* встановлюються тільки Shell. Вони дозволяють за допомогою команди *echo* отримати наступну інформацію:

- – поточні прапорці інтерпретатора (установка прапорців може бути змінена командою *set*);

– число аргументів, яке було збережено інтерпретатором при виконанні якої-небудь команди;

? – код повернення останньої виконуваної команди;

$ – числовий ідентифікатор поточного процесу PID;

! – PID останнього фонового процесу.

Арифметичні операції

Команда ***expr*** (*express* – висловлювати) обчислює вираз і записує результат до стандартного виведення даних. Елементи вираження поділяються пробілами; символи, які мають спеціальний сенс у командній мові, потрібно екранувати. Рядки, що містять спеціальні символи, укладають в апострофи. Використовуючи команду ***expr***, можна виконувати додавання, віднімання, множення, ділення, взяття залишку, зіставлення символів і т. д.

Приклад. Додавання, віднімання:

b = 190

a = `expr 200 - $b`

де ` – зворотна лапка (ліва верхня клавіша). Множення *, ділення /, взяття залишку **%**:

d = `expr $a + 125*** «*» ***10`

c = `expr $d % 13`

Тут знак множення закладається в подвійні лапки, щоб інтерпретатор не сприймав його як метасимвол. У другому рядку змінній *c* присвоюється значення залишку від ділення змінної *d* на *13*.

Вбудовані команди

Вбудовані команди є частиною інтерпретатора і не вимагають для свого виконання проведення послідовного пошуку файлу команди і створення нових процесів. Вбудовані команди:

cd [dir] – призначення поточного каталогу;

exec [cmd [arg ...]] <ім'я файлу> – виконання команди, заданої аргументами ***cmd*** і ***arg***, шляхом виклику відповідного виконуваного файлу;

umask [-o | -s] [nnn] – встановлює маску створення файлу (маску режимів доступу створюваного файлу, рівну восьмирічному числу *nnn*: 3 вісімкових цифри для користувача, групи та інших). Якщо аргумент *nnn* відсутій, то команда повідомляє поточне значення маски. При наявності прапора *-o* маска виводиться у восьмирічному вигляді, при наявності прапора *-s* – у символьному вигляді;

set, unset – режим роботи інтерпретатора, присвоювання значень параметрам;

eval [-arg] – обчислення і виконання команди;

sh <filename.sh> – виконання командного файлу *filename.sh*;

exit [n] – призводить до припинення виконання програми, повертає код повернення, рівний нулю, в програму, що викликала;

trap [cmd] [cond] – перехоплення сигналів переривання, де: *cmd* – виконувана команда; *cond = 0* або *EXIT* – у цьому випадку команда *cmd* виконується при завершенні інтерпретатора; *cond = ERR* – команда *cmd* виконується при виявленні помилки; *cond – символьне або числове позначення сигналу*, в цьому випадку команда *cmd* виконується при приході цього сигналу;

export [name [= word] ...] – включення в середовище. Команда *export* оголошує, що змінні *name* будуть включатися в середовище всіх команд, які будуть згодом викликатися;

wait [n] – очікування завершення процесу. Команда без аргументів очікує завершення процесів, запущених синхронно. Якщо вказано числовий аргумент *n*, то *wait* очікує фоновий процес з номером *n*;

read name – команда вводить рядок зі стандартного вводу і привласнює прочитані слова змінним, заданим аргументами *name*.

Приклад. Нехай є shell-файл *data*, що містить дві команди:

echo -n «Please write down your name:»

read name

Якщо викликати файл на виконання, ввівши його ім’я, то на екрані з’явиться повідомлення:

Please write down your name:

Програма очікує введення з клавіатури (в даному випадку – *ім’я* користувача). Після введення імені та натискання клавіші Enter команда виконається і на наступному рядку з’явиться знак – запрошення.

Управління програмами

Команди ***true*** і ***false*** служать для встановлення необхідного коду завершення процесу: ***true*** – успішне завершення, код завершення 0; ***false*** – неуспішне завершення, код може мати кілька значень, за допомогою яких визначається причина неуспіху завершення. Коди завершення команд використовуються для прийняття рішення про подальші дії в операторах циклу ***while*** і ***until*** та в умовному операторі ***if***. Багато команд LINUX виробляють код завершення тільки для підтримки цих операторів.

Умовний оператор if перевіряє значення виразу. Якщо воно дорівнює *true*, Shell виконує наступний за *if* оператор, якщо *false*, то наступний оператор пропускається. Формат оператора *if*:

```
if <умова>
   then
      list1
   else
     list2
fi
```

Команда ***test*** (перевірити) використовується з умовним оператором *if* і операторами циклів. Дії при цьому залежать від коду повернення *test*. *Test* проводить аналіз файлів, числових значень, ланцюжків символів. Нульовий код видається, якщо при перевірці результат позитивний, ненульовий код при негативному результаті перевірки.

У разі аналізу файлів синтаксис команди наступний:

test [-arwfds] file

де

-a – файл існує;

-r – файл існує і його можна прочитати (код завершення 0);

-w – файл існує і в нього можна записувати;

-f – файл існує і не є каталогом;

-d – файл існує і є каталогом;

-s – розмір файлу відрізняється від нуля.

При аналізі числових значень команда ***test*** перевіряє, чи істинне дане відношення, наприклад, чи рівні A і B. Порівняння виконується в форматі:

	-eq	A = B
	-ne	A <> B
test A **-ge** B	еквівалентно	A> = B
	-le	A <= B
	-gt	A> B
	-lt	A <B

Відношення зліва використовуються для числових даних, праворуч – для символів.

Крім команди **test** є ще деякі засоби для перевірки:

! – операція заперечення інвертує значення виразу;

o – операція «АБО» (or) дає значення *true*, якщо один з операндів має значення *true*;

a – операція «І» (and) дає значення *true*, якщо обидва операнди мають значення *true*.

Розгалуження з багатьох напрямків case. Команда ***case*** забезпечує розгалуження з багатьох напрямків залежно від значень аргументів команди. Формат:

case <string> in
s1) <list1>;;
s2) <list2>;;
.
.
.
sn) <listn>;;
****) <List>***
esac

Тут ***list1***, ***list2*** ... ***listn*** – список команд. Проводиться порівняння шаблону ***string*** з шаблонами ***s1, s2 ... sk ... sn***. При збігу виконується список команд, що стоїть між поточним шаблоном ***sk*** і відповідними знаками ***;;***

Наприклад:

echo -n 'Please, write down your age'

read age
case $age in
test $age -le 20) echo 'you are so young';;
test $age -le 40) echo 'you are still young';;
test $age -le 70) echo 'you are too young';;
****) Echo 'Please, write down once more'***
esac

В кінці тексту поміщена зірочка * на випадок декількох невдалих спроб ввести числа.

Цикли

Оператор циклу з умовою ***while*** *true* і ***while*** false. Команда ***while*** (поки) формує цикли, які виконуються до тих пір, поки команда **while** визначає значення наступного за ним виразу як *true* або *false*. Формат оператора циклу з умовою **while** *true*:

while ***list1***
do
list2
done

Тут *list1* і *list2* – списки команд. ***While*** перевіряє код повернення списку команд, що стоять після ***while***, і якщо його значення дорівнює 0, то виконуються команди, які стоять між ***do*** і ***done***. Оператор циклу з умовою **while** *false* має формат:

until ***list1***
do
list2
done

На відміну від попереднього випадку умовою виконання команд між ***do*** і ***done*** є ненульове значення повернення. Програмний цикл може бути розміщений усередині іншого циклу (вкладений цикл). Оператор ***break*** перериває найближчий до нього цикл. Якщо в програму ввести оператор ***break*** з рівнем 2 (***break 2***), то це забезпечить вихід за межі двох циклів і завершення програми.

Оператор ***continue*** передає управління найближчого в циклі оператору ***while***.

Оператор циклу з перерахуванням ***for***:

for name in [wordlist]
do
list
done

де ***name*** – змінна; ***wordlist*** – послідовність слів; ***list*** – список команд. Змінна ***name*** отримує значення першого слова послідовності ***wordlist***, після цього виконується список команд, що стоїть між ***do*** і ***done***. Потім ***name*** отримує значення другого слова ***wordlist*** і знову виконується список ***list***. Виконання зупиниться після того, як закінчиться список ***wordlist***.

Завдання до виконання

1. Вивчити організацію умовного виконання командного рядка, угруповання команд у командному рядку.
2. Вивчити використання змінних shell.
3. Ознайомитися з організацією командних файлів: передачею параметрів, уведенням значень, умовними розгалуженнями і циклами.
4. Вивчити арифметичні обчислення в shell.
5. Написати сценарій для оболонки bash. Сценарій шукає у заданому каталозі і його підкаталогах усі файли, які містять послідовність байт, що відповідає шаблону пошуку, і формує їх список. Список повинен містити ім'я файлу, повний шлях до каталогу, тип файлу, розмір файлу у байтах, і рядок, що відповідає шаблону.

Контрольні питання

1. Яке значення мають shell-файли?
2. Як створити shell-файл і зробити його виконуваним?
3. Які типи змінних використовуються в shell-файлах?
4. У чому полягає аналіз ланцюжка символів?
5. Які вбудовані команди використовуються в shell-файлах?
6. Як здійснюється управління програмами?
7. Назвіть оператори створення циклів.

Робота № 6. Утиліта awk. Змінні, записи, поля, масиви

Мета роботи: оволодіння практичними навичками професійної роботи з утилітою AWK.

Короткі теоретичні відомості

Утиліта AWK була створена в 1977 р, американськими авторами Alfred V.Aho, Brian W.Kernighan і Peter J.Weinberger і призначена для простих, механічних та обчислювальних маніпуляцій над даними. Досить нескладні операції часто необхідно виконати над цілими пакетами файлів, а писати для цього програму на одному із стандартних мов програмування є виснажливою і, як правило, не дуже простою справою. Оптимальне рішення проблеми – використання спеціальної утиліти AWK, що включає в себе не громіздку і зручну мову програмування, яка дозволяє вирішувати завдання обробки даних за допомогою коротких програм, що складаються з двох-трьох рядків.

AWK сканує input (стандартний або вказаний набір файлів) і над рядками, що задовольняють заданому зразку, виконує вказані дії. Рядок може містити максимально до 256 символів.

Формат:

awk [-Fc] [-f file] [files]
awk [-Fc] [prog] [files]

prog – програма, виду: ‘зразок $ {$ дія $} $’;
file – файл з AWK-програмою:
зразок {дія}
зразок {дія}
...
files – файли, призначені для AWK-обробки.
-Fc – встановлює роздільник полів в «***c***»

Загальна структура AWK-програми

Мова програмування AWK допускає використання: полів; змінних (стандартних, масивів); арифметичних виразів;

зразків такого вигляду: регулярний вираз; вираз відносин; комбінація зразків; слова BEGIN і END.

Дію в AWK визначає: послідовність пропозицій, розділених ‘’;’’ або ‘’\ n’’ (новий рядок).

Пропозицію визначають: виведення(друк); присвоювання; вбудована функція; керуюча структура.

Поля. Кожен сканований рядок input розглядається як поля, що розділені розділовими символами (за замовчуванням – пробіл).

На поля можна посилатися з AWK програми наступним чином:

$1 – перше поле;

$2 – друге поле і т. д.;

$0 – посилається на весь рядок цілком.

Рядок може містити максимально до 100 полів.

Змінні

Значення змінних

Змінні можуть інтерпретуватися як числові або строкові. Вони приймають значення в залежності від контексту, наприклад:

x = 1, x – сприймається як число;

x = «», x – рядок;

x + «abc» – результат операції інтерпретується як число незалежно від того, чи було x числом або рядком. Якщо рядок не може бути інтерпретований як число («abc»), то його значення стає 0.

Рядок може містити максимально до 256 символів.

Змінні поля

Посилання на поля $1, $2, ... можуть інтерпретуватися як змінні, наприклад:

$1 = «3» +$2 – перше поле приймає значення другого поля, збільшеного на 3.

$(i+1) – інтерпретується як поле, номер якого залежить від значення змінної i.

Масиви

Допускається використання масивів. Масиви не декларуються, а приймають значення з контексту, наприклад:

x[NR]=$0 – елементу масиву x, індексованому NR, присвоюється рядок (NR – number of records – номер запису);

x[«apple»] – елементи масиву можуть індексуватися не числовим значенням, тобто рядком.

Арифметичні вирази

Вираз включає: змінну; число; рядок; вбудовану функцію.

Арифметичні вирази мають вигляд:

Вираз <Операція> Вираз

<Операція>: «+», «-», «*», «/», «%»

Зразки / patterns /

Регулярний вираз

Для здійснення пошуку в AWK мовою допускається використання регулярних виразів, зазначених в описі. Доповнення до використання регулярних виразів, які припускаються в AWK-мові:

«()» – дужки допускаються для групування;

«|» – вказівка альтернативи «або»;

«+» – плюс, що стоїть за регулярним виразом означає будь-яку послідовність входжень цього виразу, починаючи з 1;

«?» – знак питання за регулярним виразом означає 0 або 1 входжень цього виразу;

[A-Z] – допускається скорочена форма запису для рангів ASCII символів;

Встановлений порядок виконання операторів на одному рівні дужок: «[] * +? конкатенація |».

Вираз відносин

Вираз відносин може бути двох типів:

<Вираз> <Приналежність> <Вираз>

<Вираз> <Лог. Операція> <Вираз>

Приналежність: ~ – Міститься; ! ~ – Не міститься.

Лог. Операція: <, <=, ==,! =,> =,>.

Комбінація зразків

Допускається логічна комбінація зразків з використанням наступних знаків: || – ‘’або’’, && – ‘’ і ‘’, ! - ‘’ не «.

BEGIN і END

Зразок BEGIN вказує на початок input або на ті дії, які повинні бути виконані до якого б то не було аналізу рядків.

Зразок END вказує на кінець input або на ті дії, які повинні бути виконані після обробки всіх рядків.

Наприклад:

BEGIN {FS = «:»} – встановлює роздільник полів в «:» до початку обробки рядків. Еквівалентно опції «-F:» при AWK виклику.

END {print NR} – друкує номер останнього рядка input, тобто кількість оброблених вхідних рядків.

Дії

Виведення (Друк)

Формат оператора друку:

PRINT [<список виразів>] [> <вираження1>]

Якщо в списку вираження знаходяться через кому, то значення цих виразів виводяться на output (друкуються) через символ-роздільник OFS (за замовчуванням пробіл). Якщо ж вирази стоять через пробіл, то на друку відбувається їх конкатенація.

Значення *<вираження1>* розглядається як ім'я файлу. Саме його присутність означає друк у файл. Якщо замість '' > '' стоїть '' >> '', то це означає додавання до вже існуючого файлу. Можна використовувати в одній програмі максимально до 10 output файлів.

Оператор форматного друку:

PRINTF формат [, список виразів] [> вираження1]

формат: символьний рядок у подвійних лапках. Ідентичний формату, використовуваному в функції printf у мові *C*. Формат може містити:

звичайні символи, вони копіюються на output;

esc-послідовності, що представляють неграфічні символи, наприклад, «\n» – новий рядок;

специфікації для виведення аргументів, вони слідують після символу ''%''. Число специфікацій має дорівнювати числу аргументів (якщо воно менше числа аргументів, то зайві аргументи ігноруються, якщо ж більше – то це помилка).

Присвоєння

Оператор присвоювання має вигляд:

<Змінна> = <вираз>

Початкове значення змінної 0 або ‘’ ‘’ (пропуск). Допускаються інші типи привласнення, відповідно до мови С: «+ =», «- =», «* =», «/ =», «% =».

<Змінна> ++, ++ <змінна> – збільшення значення змінної на 1.

<Змінна> -, - <змінна> – зменшення значення змінної на 1.

Вбудовані функції

length(arg) – функція довжини *arg*. Якщо *arg* не вказано, то видає довжину поточного рядка.

exp(), log(), sqrt() – математичні функції експонента, логарифм і квадратний корінь.

int() – функція цілої частини числа.

substr (s, m, n) – повертає підрядок рядка *s*, починаючи з позиції *m*, всього *n* символів.

index(s, t) – повертає початкову позицію підрядка *t* в рядку *s*. (Або 0, якщо *t* в *s* не міститься.)

sprintf(fmt, exp1, exp2, ...) – здійснює форматований друк (виведення) в рядок, ідентично PRINTF.

split(s, array, sep) – поміщає поля рядка *s* в масив *array* і повертає число заповнених елементів масиву. Якщо вказано *sep*, то при аналізі рядка він розуміється як роздільник.

Завдання до виконання

Створити в домашньому каталозі файл з інформацією про співробітників. Дані розташувати в табличному вигляді в наступній послідовності: прізвище, ім’я, назва структурного підрозділу (СП), оклад, рік народження. Як роздільник полів використовувати табуляцію.

Використовуючи команду awk:

1. Відобразити на екрані всю інформацію про співробітників із зазначеного файлу в тій послідовності, в якій вона представлена в файлі.

2. Відобразити на екрані всю інформацію про співробітників із зазначеного файлу в зворотній послідовності в порівнянні з тією, в якій вона представлена в файлі.

3. Показати тільки прізвища, імена співробітників з нумерацією кожного запису. Інформацію в рядку виводити з використанням пробілу.

4. Показати номери структурних підрозділів, прізвища співробітників. Інформацію в рядку виводити з використанням табуляції. Отримані дані перенаправити в інший файл домашнього каталогу, відсортовані по першій колонці дані відобразити поекранно.

5. Показати всі дані про конкретного співробітника. Інформацію в рядку виводити з використанням подвійної табуляції.

6. Показати прізвища співробітників конкретного структурного підрозділу.

7. Показати прізвища, назви СП співробітників одного року народження.

8. Показати назви СП, в яких є співробітники старше 1985 року народження.

9. Визначити сумарне значення по полю оклад працівників даної організації.

10. Визначити сумарне значення по полю оклад співробітників конкретного СП.

11. Показати оклад, прізвище співробітника, що має максимальний оклад. При виведенні даних використовувати наступний формат: десятисимвольне поле для відображення прізвища, цілочисельне чотирисимвольне поле для відображення окладу.

12. Завдання виконати:

а) в командному рядку;

б) шляхом написання спеціальної програми для сканування даних і формування звіту

Контрольні питання

1. Призначення утиліти AWK.
2. Вбудована мова програмування утиліти AWK та її основні конструкції.
3. Формат та параметри утиліти AWK.
4. Загальна структура AWK-програми.

Робота № 7. Команда ifconfig. Файли конфігурації мережі

Мета роботи: освоїти правила конфігурування мережевих інтерфейсів використовуючи утиліту *ifconfig*.

Короткі теоретичні відомості

Утиліта *ifconfig* дозволяє переглянути поточну конфігурацію адресів TCP/IP для всіх встановлених на даному комп'ютері мережевих адаптерів і комутованих з'єднань, з її допомогою можна визначити IP-адресу вашого комп'ютера.

Утиліта *ifconfig* має дуже простий синтаксис при виклику без параметрів, вона поверне список підключених до системи мережевих інтерфейсів та їх характеристики, наприклад, ір-адресу, адресу шлюзу, розмір пакета, частоту для бездротових мереж та інші параметри. Якщо вказати інтерфейс і потрібні команди, то можна змінювати різні настройки інтерфейсу.

У більшості UNIX-подібних систем (у тому числі і Linux) команда *ifconfig* має наступний синтаксис:

ifconfig інтерфейс [сімейство] адреса опції. . .

Так, наприклад, наступна команда:

ifconfig eno1 192.168.0.100 netmask 255.255.255.0 up

призначає IP-адресу *192.168.0.100* для пристрою *eno1*, яку використовує відповідна мережева карта, призначає підмережу для даного вузла за допомогою опції *netmask* і задіє – опцією *up*. В даному випадку параметр [*сімейство*] не задіяний, так як за замовчуванням використовується – протокол IPv4. Для одного інтерфейсу можна задавати декілька протоколів, але конфігурувати їх потрібно окремо. Як адресу можна також використовувати й імена вузлів. Опція *up* в даному прикладі задана явно, хоча при призначенні адреси її активація/включення відбувається за замовчуванням. Для відключення використовується опція *down*.

Для команди *ifconfig* існує багато опцій і багато з них вимагають додаткових параметрів, які вводяться відразу після імені опції через символ пропуску. Ось деякі з них:

add – додає адресу для мережевого інтерфейсу.

del – видаляє адресу для мережевого інтерфейсу.

io_addr – встановлює початкову адресу для введення-виведення мережевого пристрою.

bootproto – задає спосіб отримання IP-адреси. Як параметри використовуються *static* або *dhcp*.

media – задає фізичний порт для використання мережним пристроєм. Зазвичай в якості значень використовуються 10base2 – коаксіальний Ethernet-кабель, 10baseT – кручена пара до 10 Мбіт/сек або AUX – зовнішній передавач.

up – включає вказаний мережевий інтерфейс або перезапускає його, якщо він був до цього відключений опцією *down*.

down – відключає мережевий інтерфейс, при цьому вся інформація про маршрути, пов'язані з цим інтерфейсом, зберігається.

netmask – задає підмережу за допомогою маски.

pointopoint – використовується для організації point-to-point з'єднань, як параметр передається адреса.

broadcast – задає широкомовну адресу.

irq – задає номер переривання для пристрою мережевого інтерфейсу.

metric – зазвичай використовується для завдання метричного значення в запису таблиці маршрутизації.

mtu – задає максимальну довжину переданого пакета (MTU), за замовчуванням дорівнює 1500 для Ethernet і 296 для SLIP

arp – дозволяє використовувати протокол ARP для пошуку фізичної адреси пристрою в мережі. Включений за замовчуванням для широкомовних мереж. Про вимкнений ARP говорить прапор NOARP у виведенні *ifconfig*.

-arp – відключає використання ARP для обраного інтерфейсу/пристрою.

promisc – змушує обраний інтерфейс отримувати всі пакети, незалежно від того, призначені вони для нього чи ні. Корисно для аналізу мережевого трафіку і виявлення проблем з мережею. При цьому зручно використовувати утиліту *tcpdump*. При активації режиму *promiscuous* у виведенні *ifconfig* присутній прапор PROMISC для даного інтерфейсу.

-promisc – відключає режим promiscuous для мережевого інтерфейсу.

allmulti – включає режим multicast, дозволяючи multicast-адреси. Цей режим дозволяє звертатися до пристроїв, які можуть перебувати в різних підмережах – дуже корисно при організації голосового зв'язку і конференцій. Може не підтримуватися мережевою картою.

hw – задає апаратний (MAC) для мережевого пристрою. Необхідно також вказувати клас пристрою: ether – для Ethernet, netrom – для AMPR NET / ROM, ax25 – для AMPR AX.25. Може не підтримуватися драйвером пристрою.

-allmulti – забороняє режим multicast.

Приклади застосування ifconfig

Включення і відключення мережевого інтерфейсу на прикладі eth0 і eno1:

ifconfig eth0 up

ifconfig eno1 down

Зміна параметра MTU для eth0:

ifconfig eth0 mtu 1000

Завдання MAC-адреси для eth0:

ifconfig eth0 hw ether AA:BB:CC:DD:EE:FF

Налаштування PLIP-інтерфейсу для зв'язку комп'ютера з IP-адресою 10.24.205.18 з комп'ютером, у якого IP-адреса 10.24.105.20:

ifconfig plip0 10.24.205.18 pointopoint 10.24.105.20

Призначення IP-адреси мережевої карти eth0:

ifconfig eth0 10.24.205.18

Зняття адреси з eth0:

ifconfig eth0 del 10.24.205.18

Завдання IP-адреси, підмережі і широкомовної адреси для eth0:

ifconfig eth0 10.24.205.18 netmask 255.255.255.0 broadcast 10.24.205.10

Це найбільш корисні опції утиліти *ifconfig* для налаштування мережевих інтерфейсів у Linux, для отримання додаткової інформації про використання утиліти *ifconfig* – використовуйте «man ifconfig» за допомогою терміналу.

Завдання до виконання

1. Ознайомитися з довідковою інформацією утиліти *ifconfig* .

2. Як називаються мережеві інтерфейси в Linux?

3. Який файл використовується для налаштування мережевих параметрів?

Контрольні питання

1. Як за допомогою команди *ifconfig* вивести параметри тільки одного мережевого інтерфейсу (наприклад, eth1), а не всіх?

2. Як змінити IP-адресу в командному рядку?

3. Як додати новий псевдонім у мережевий інтерфейс?

Робота № 8. Команди route, ip. Таблиці маршрутизації

Мета роботи: освоїти налаштування мережевої маршрутизації використовуючи утиліти *route, ip*.

Короткі теоретичні відомості

Для визначення і завдання маршрутів у мережах існують динамічна і статична маршрутизації. У першому випадку маршрути задаються спеціальним демоном маршрутизації, який модифікує відповідним чином таблицю маршрутизації ядра. У другому випадку маршрути задаються адміністратором / користувачем за допомогою команди *route*. Маршрути, задані командою *route* не змінюються, навіть якщо включена динамічна маршрутизація.

Процеси маршрутизації здійснюються на мережевому рівні. Для кожного пакету проводиться порівняння його цільової IP-адреси з записами в таблиці маршрутизації. Коли виявляється хоча б часткова відповідність з одним із шлюзів у таблиці, пакет направляється до наступного вузла (шлюзу), відповідному знайденому маршруту. І тут може виникати кілька ситуацій:

перша – коли, наприклад, пакет адресується комп'ютеру, що знаходиться в тій же мережі, що й джерело пакета, а точніше сказати – його відправник. У даній ситуації для такого пакета наступним шлюзом є один з локальних інтерфейсів і він (пакет) відправляється відразу до адресата. Такі «явні» і «короткі» шлюзи зазвичай задаються під час конфігурації мережевих інтерфейсів – командою *ifconfig*;

друга – коли адреса призначення пакету не відповідає жодному шлюзу в таблиці маршрутизації. В такому випадку, щоб уникнути колізій у мережі і її надмірного навантаження, повинен бути задіяний шлюз за замовчуванням. Іншими словами, це такий маршрут, який вказує системному ядру: всі інші пакети (без відповідників у таблиці маршрутів) направляй сюди. Якщо шлюз за замовчуванням не буде передбачено, то стороні, що відправляє, надсилається повідомлення про недосяжність мережі або вузла.

Як правило, локальні мережі мають єдиний шлюз у зовнішнє середовище, наприклад в Інтернет. У свою чергу, в мережі Інтернет таких «стандартних маршрутів» не існує.

Синтаксис і основні опції утиліти route

Основне призначення команди *route* – додавання і видалення мережевих маршрутів для системного ядра, а також перегляд вмісту таблиці маршрутизації. Ця команда, хоча і працює в різних UNIX-подібних системах однаково, проте різко відрізняється синтаксис в залежності від використовуваної системи.

У загальному випадку прототипом команди *route* є такий запис:

route add [-net|-host] <IP/Net> netmask gw <Gateway IP> dev <Int>X

Наприклад:

route add -net 127.0.0.0 netmask 255.0.0.0 metric 1024 dev lo

Ця команда додасть шлюз зі зворотним зв'язком через віртуальне пристрій *lo*, який використовується для цієї мети в Linux-системах. Опції *-net* і *-host* використовуються для вказівки адреси, що характеризує або мережу, або вузол відповідно як пункти призначення. Для визначення підмережі служить опція *netmask*, для завдання пріоритету шлюзу – опція *metric*. Мережевий інтерфейс позначається опцією *dev*. Крім описаних вище для команди *route* також існують і інші використовувані нею опції, які наведені нижче:

del – видалення маршруту;

gw – шлюз, через який повинні досягатися мережа або вузол. Задається у вигляді імені вузла або точкового запису адреси;

mss – встановлює значення MTU (максимальну величину пакета) в байтах;

window – встановлює розмір TCP-вікна для заданого шлюзу в байтах. Зазвичай використовується в мережах AX.25;

irtt – встановлює проміжок часу відгуку для TCP-з'єднань по даному маршруту в мілісекундах;

reject – задає блокуючий маршрут, який повинен призводити до зупинки процедури пошуку маршрутів. Корисно при приховуванні мереж для використання в них шлюз за замовчуванням;

-F – змушує працювати з таблицею маршрутизації ядра. Ця опція в більшості систем використовується за замовчуванням, тому часто опускається;

-C – змушує працювати з кешем маршрутизації ядра;

-v – включає докладний режим роботи команди route;

-n – використання числового формату адрес замість спроб визначення символьних найменувань вузлів. Можна використовувати в разі визначення проблем із з'єднаннями до DNS;

-e – використовувати формат виведення команди netstat для відображення вмісту таблиці маршрутів. Опція *-ee* згенерує найдокладніший звіт з повними найменуваннями параметрів таблиці маршрутів.

Приклади використання.

Визначити маршрут до мережі, яка повинна бути досягнута через мережевий інтерфейс eth0:

ifconfig eth0 nodeone netmask 255.255.255.0

route add -net 192.168.1.0

Тут для команди *route* не вказується сам інтерфейс, оскільки передбачається, що вузлу *nodeone* відповідає адреса 192.168.1.2. Далі *route* «дізнається», що маршрут потрібно прокласти саме через eth0 завдяки тому, що системне ядро аналізує всі доступні інтерфейси на предмет їх конфігурації і порівнює адресу пункту призначення з мережевою частиною мережевих (сконфігурованих) інтерфейсів. У даному випадку ядро виявляє, що eth0 – той інтерфейс (з адресою 192.168.1.2), якому відповідає кінцева адреса, тобто 192.168.1.0.

Завдання шлюзу за замовчуванням:

route add default gw 192.168.1.1 eth0

Доступ в локальну мережу Ethernet через мережевий інтерфейс eth0:

route add -net 192.168.10.0 netmask 255.255.255.0 eth0

Тут *192.168.10.0* – мережа, до якої потрібно встановити доступ (маршрут).

Виведення вмісту таблиці маршрутів ядра здійснюється командою *route* без параметрів, для докладних результатів використовується опція *-ee*.

Також можна використовувати скорочений запис для завдання маски підмережі:

route add -net 192.168.10.0/24 eth0

Слід зазначити, що шлюзи, встановлені командою *route*, будуть існувати до перезавантаження системи. Для їх використання на постійній основі необхідно потрібні команди прописати в файлі. В Ubuntu це */etc/network/interfaces*.

Синтаксис і основні опції утиліти ip

Команда *ip* – це потужний інструмент для налаштування мережевих інтерфейсів Linux. Її можна використовувати для підвищення або зниження інтерфейсів, призначення та видалення адрес, маршрутів, управління кешем ARP і багато чого іншого. Утиліта *ip* є частиною пакета *iproute2*, який встановлюється у всіх сучасних дистрибутивах Linux.

Синтаксис команди *ip* наступний:

ip [OPTIONS] OBJECT { COMMAND | help }

OBJECT – це тип об'єкта, яким необхідно керувати. Найбільш часто використовувані об'єкти (або підкоманди):

link (l) – відображення або зміна мережевих інтерфейсів;

address (a) – відображення і зміна IP-адрес;

route (r) – показати і змінити таблицю маршрутизації;

neigh (n) – відображати і маніпулювати сусідніми об'єктами (таблиця ARP).

Об'єкт може бути написаний у повній або скороченій (короткій) формі. Для відображення списку команд і аргументів для кожного типу об'єкта використовується *ip OBJECT help*.

Конфігурації, встановлені за допомогою команди *ip*, не є постійними. Після перезавантаження системи всі зміни будуть втрачені. Для постійних налаштувань потрібно відредагувати дистрибутивні файли конфігурації або додати команди в скрипт запуску.

Приклади використання.

Для відображення списку всіх мережевих інтерфейсів і пов'язаних з ними IP-адрес введіть наступну команду:

ip addr show

Для отримання інформації про конкретний мережевий інтерфейс використовуйте ім'я пристрою *ip addr show dev*. Наприклад, щоб запросити *eth0*, введіть:

ip addr show dev eth0

Щоб додати адресу 192.168.101.12 з мережевою маскою 24 на пристрій eth0, введіть:

ip address add 192.168.101.12/24 dev eth0

За допомогою утиліти *ip* ви можете призначити декілька адрес одному інтерфейсу, наприклад:

ip address add 192.168.113.241/24 dev eth0

ip address add 192.168.101.12/24 dev eth0

Для видалення адреси 192.168.101.12/24 з пристрою eth0 введіть:

ip address del 192.168.101.12/24 dev eth0

Щоб відобразити список усіх мережевих інтерфейсів, введіть наступну команду:

ip link show

На відміну від *ip addr show*, *ip link show* не буде друкувати інформацію про IP-адреси, пов'язані з пристроєм.

Для включення або виключення інтерфейсу використовуйте *ip link set dev DEVICE* і бажаний стан. Наприклад, щоб вивести інтерфейс *eth0* в онлайн, ви повинні набрати:

ip link set eth0 up

І вивести якщо офлайн:

ip link set eth0 down

Для призначення, видалення і відображення таблиці маршрутизації ядра використовуйте об'єкт *route*. Найбільш часто використовувані команди при роботі з об'єктами маршрутів *list*, *add* і *del*.

Наприклад, щоб показати таблицю маршрутизації використовуйте список записів маршруту ядра, використовуючи одну з наступних команд:

ip route

ip route list

ip route list SELECTOR

Наприклад, щоб відобразити тільки маршрутизацію для конкретної мережі 172.17.0.0/16, введіть:

ip r list 172.17.0.0/16

Щоб додати новий запис у таблицю маршрутизації, використовуйте команду *route add*, а потім ім'я мережі або пристрою. Наприклад, щоб додати маршрут до 192.168.113.0/24 через шлюз в 192.168.113.1:

ip route add 192.168.113.0/24 via 192.168.113.1

Щоб додати маршрут за замовчуванням, використовуйте ключове слово *default*. Наступна команда додасть маршрут за замовчуванням через локальний шлюз *192.168.113.1*, який може бути досягнутий на пристрої *eth0*:

ip route add default via 192.168.113.1 dev eth0

Наступна команда видалить маршрут за замовчуванням:

ip route del default

Для отримання додаткової інформації про інші параметри *ip* відвідайте сторінку допомоги та керівництва команди *ip.*

Завдання до виконання

1. Ознайомитися з довідковою інформацією утиліт *route, ip*.
2. Що таке таблиця маршрутизації?
3. Що таке шлюз?

Контрольні питання

1. Як змінити статус мережевого інтерфейсу?
2. Як добавити IP-адресу в інтерфейс?
3. Як відобразити таблицю IP маршрутизації?
4. Як змінити записи таблиці IP маршрутизації?

Робота № 9. Команда iptables. Ланцюжки iptables. Мережева трансляція адрес (NAT).

Мета роботи: освоїти синтаксис команд управління фільтром пакетів в ОС Linux. Освоїти техніку роботи з трансляцією мережних адрес (NAT).

Короткі теоретичні відомості

Iptables – це брандмауер для Linux, який використовується для моніторингу вхідного і вихідного трафіку та його фільтрації відповідно до заданих користувачем правил для виключення несанкціонованого доступу до системи. За допомогою *Iptables* можна дозволити на вашому сервері рух тільки певного трафіку.

Всі дані передаються по мережі у вигляді пакетів. Ядро Linux надає інтерфейс для фільтрації пакетів вхідного і вихідного трафіку за допомогою спеціальних таблиць. Iptables – це додаток командного рядка і міжмережевий екран для Linux, яким можна користуватися для створення, підтримки працездатності та перевірки цих таблиць.

Можна створити кілька таблиць. У кожній таблиці може міститися кілька ланцюжків. Ланцюжок – це набір правил. Кожне правило визначає, що робити з пакетом, якщо він відповідає умовам. При відповідності пакета над ним виконується цільова дія (TARGET). Це може бути перевірка наступного ланцюжка або один з наступних варіантів:

ACCEPT: дозволити передачу пакета.

DROP: заборонити передачу пакета.

RETURN: пропустити поточний ланцюжок і перейти до наступного правила в ланцюжку, який його викликав.

У таблиці фільтра є три ланцюжки (набору правил):

INPUT – використовується для контролю вхідних пакетів. Можна дозволяти чи блокувати підключення по порту, протоколу або IP-адреси джерела.

FORWARD – використовується для фільтрації пакетів, що приходять на сервер, але їх потрібно переправити кудинебудь ще.

OUTPUT – використовується для фільтрації вихідних пакетів.

Для перевірки поточного стану *Iptables* використовується наступна команда:

sudo iptables -L -v

Ця команда дозволить перевірити стан поточної конфігурації *Iptables*. Опція *-L* використовується для виведення всіх правил, а опція *-v* – для більш докладної інформації.

Визначення правил

Визначення правила означає внесення його до списку (ланцюжок). Ось приклад команди *Iptables* зі звичайними опціями. Деякі з них не є обов'язковими:

sudo iptables -t <таблиця> -A <ланцюжок> -i <інтерфейс> -p <протокол (tcp / udp)> -s <джерело> --dport <номер порту> -j <цільова дія>

Опція *-A* означає «append», додавання правила в кінець ланцюжка. Дуже важливим моментом є те, що правила в ланцюжку застосовуються послідовно, і якщо пакет задовольняє умовам одного з них, наступні застосовуватися не будуть. Це завжди потрібно пам'ятати при створенні наборів правил. *Таблиця* означає таблицю, в якій редагуються ланцюжки. За замовчуванням це *filter*. *Ланцюжок* – ім'я ланцюжка, до якого ми додаємо правило. *Інтерфейс* – мережевий інтерфейс, для якого виконується фільтрація трафіку. *Протокол* вказує мережевий протокол, пакети якого ви хочете фільтрувати, *джерело* – IP-адреса відправника пакета, *номер порту* – порт на вашому сервері, для якого застосовується правило. Ці та деякі інші опції будуть розглянуті нижче при демонстрації відповідних дій. Для більш докладної інформації про команду Iptables та її опції зверніться до відповідної *man*-сторінки:

man iptables

<u>Приклад налаштування iptables</u>

Розглянемо настройку *iptables* на прикладі настройки доступу до web сервера.

1. Дозвіл трафіку на локальному вузлі

Насамперед нам потрібно обов'язково дозволити трафік через інтерфейс *loopback*, щоб зв'язок між додатками і базами даних на сервері тривав в звичайному режимі.

sudo iptables -A INPUT -i lo -j ACCEPT

Опція *-A* використовується для додавання в ланцюжок *INPUT* правила, що дозволяє всі з'єднання для інтерфейсу *lo*. Це позначення для інтерфейсу *loopback*, він використовується для всього зв'язку на локальному вузлі, наприклад, зв'язку між базою даних і веб-додатком на одній машині.

2. Дозвіл підключень до портів HTTP, SSH і SSL

Нам потрібно, щоб у звичайному режимі працювали порти HTTP (порт 80), https (порт 443), ssh (порт 22). Для цього потрібно виконати наступні команди:

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT

У них ми за допомогою опції *-p* вказали протокол, а за допомогою опції *--dport* (destination port, порт на приймаючій стороні, в даному випадку це ваш сервер) відповідний кожному протоколу порт. Тепер всі TCP-підключення з цими номерами портів будуть дозволені. Насправді для підключення по порту ssh краще дозволити не з усіх ip адрес, а тільки з певних.

3. Фільтрація пакетів за джерелом

Якщо необхідно приймати або відхиляти пакети по IP-адресі або діапазону IP-адреси джерела, можна вказати їх за допомогою опції *-s*. Наприклад, щоб приймати пакети з адреси 192.168.1.3:

sudo iptables -A INPUT -s 192.168.1.3 -j ACCEPT

замість хоста також можна вказати і повністю мережу, наприклад для вирішення підключення по *ssh* тільки з локальної мережі *192.168.1.0/24* необхідно прописати правило:

sudo iptables -A INPUT -s 192.168.1.0/24 -p tcp --dport 22 -j ACCEPT

Відхилення команд з IP-адреси задається аналогічною командою з опцією DROP:

sudo iptables -A INPUT -s 192.168.1.3 -j DROP

Якщо потрібно відхиляти або приймати пакети з діапазону IP-адрес, необхідно скористатися модулем *iprange* з опцією *-m* і за допомогою параметра *--src* вказати діапазон:

sudo iptables -A INPUT -m iprange --src-range 192.168.1.100-192.168.1.200 -j DROP

4. Заборона всього іншого трафіку

Після визначення правил обов'язково потрібно заборонити решті трафіку, з метою виключення несанкціонованого доступу до сервера через інші відкриті порти.

sudo iptables -A INPUT -j DROP

Видалення правил iptables

Якщо потрібно видалити всі правила і прописати все заново з чистого аркуша, можна скористатися командою очищення (flush):

sudo iptables -F

Вона видаляє всі існуючі правила. Якщо потрібно видалити конкретне правило, скористайтеся опцією *-D*. Спочатку за допомогою опції *-line-numbers* виведіть пронумерований список усіх правил:

sudo iptables -L --line-numbers

Для видалення правила потрібно вказати ланцюжок і номер у списку. В нашому випадку це ланцюжок INPUT і номер 3:

sudo iptables -D INPUT 3

Налаштування NAT (MASQUERADE)

Більшість організацій отримує від своїх інтернет-провайдерів обмежену кількість публічно доступних IP-адрес, тому адміністратору потрібно розділяти доступ до Інтернету, не видаючи кожному вузлу мережі публічну IP-адресу. Для доступу до внутрішніх і зовнішніх мережних служб вузли локальної мережі використовують внутрішню (приватну) IP-адресу. Граничні маршрутизатори (міжмережеві екрани) можуть отримувати вхідні з'єднання з Інтернету і передавати їх потрібним вузлам локальної мережі і навпаки, направ-

ляти вихідні з'єднання вузлів віддаленим інтернет-службам. Таке перенаправлення мережевого трафіку може бути небезпечним, особливо з огляду на доступність сучасних засобів злому, які здійснюють підміну внутрішніх IP-адрес і таким чином маскують машину зловмисника під вузол вашої локальної мережі. Щоб цього не сталося, в *iptables* існують політики маршрутизації і перенаправлення, які можна застосовувати для виключення неправомірного використання мережевих ресурсів. Політика FORWARD дозволяє адміністратору контролювати маршрутизацію пакетів у локальній мережі. Наприклад, щоб дозволити перенаправлення всієї локальної мережі, можна встановити такі правила (в прикладі передбачається, що внутрішня IP-адреса брандмауера/шлюзу призначена на інтерфейсі eth1):

sudo iptables -A FORWARD -i eth1 -j ACCEPT

sudo iptables -A FORWARD -o eth1 -j ACCEPT

Ці правила надають системам за шлюзом доступ до внутрішньої мережі. Шлюз направляє пакети від вузла локальної мережі до вузла призначення і навпаки (опції *-o* і *-i*), передаючи їх через пристрій eth1.

За замовчуванням в більшості дистрибутивів Linux перенаправлення відключено. Щоб його включити, потрібно внести зміни в файл конфігурації /etc/sysctl.conf. Знайдіть там такий рядок net.ipv4.ip_forward = 0 і змініть 0 на 1: *net.ipv4.ip_forward = 1.*

Дозвіл перенаправлення пакетів внутрішнім інтерфейсом брандмауера дозволяє вузлам локальної мережі зв'язуватися один з одним, але у них не буде доступу в Інтернет. Щоб забезпечити вузлам локальної мережі з приватними IP-адресами можливість зв'язку з зовнішніми публічними мережами, потрібно налаштувати на брандмауері трансляцію мережевих адрес (NAT). Створимо правило:

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

У цьому правилі використовується таблиця NAT, тому вона вказується в явному вигляді *(-t nat)* і вказаний вбудований

ланцюжок цієї таблиці POSTROUTING для зовнішнього мережевого інтерфейсу *eth0 (-o eth0)*. POSTROUTING дозволяє змінювати пакети, коли вони виходять з зовнішнього інтерфейсу шлюзу. Цільову дію -j MASQUERADE замінює (маскує) приватна IP-адреса вузла зовнішньою IP-адресою шлюзу. Це називається IP-маскарадінг.

Перекидання портів

Якщо у внутрішній мережі є сервер, який повинен бути доступний ззовні, можна скористатися цільовою дією *-j DNAT* ланцюжка PREROUTING в таблиці NAT і вказати IP-адресу та порт призначення, на які будуть перенаправлятися вхідні пакети, що запитують з'єднання з вашою внутрішньою службою. Наприклад, якщо потрібно перенаправляти вхідні HTTP-запити по порту 80 на виділений HTTP-сервер Apache за адресою 172.31.0.23, виконайте команду:

sudo iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to 172.31.0.23:80

Дане правило визначає використання вбудованого ланцюжка PREROUTING в таблиці NAT для перенаправлення вхідних HTTP-запитів на порт 80 зазначеного IP-адресу 172.31.0.23. Такий спосіб трансляції мережевих адрес називається перенаправленням або перекиданням портів.

Збереження змін в iptables

Створені нами правила збережені в оперативній пам'яті. Це означає, що при перезавантаженні нам потрібно буде заново їх визначити. Щоб зберегти їх після перезавантаження, потрібно виконати наступну команду:

<u>В Ubuntu / Debian</u>

*sudo iptables-save> /etc/iptables.rule*s

<u>В Centos / Redhat</u>

iptables-save> /etc/sysconfig/iptables

Команда зберігає поточні правила в файл конфігурації системи, який використовується для переналаштування таблиць під час перезавантаження. Цю команду потрібно виконувати після кожної зміни правил. Для відключення брандмауера просто очистіть всі правила і збережіть зміни:

sudo iptables -F
sudo /sbin/iptables-save

Відновлення налаштувань

Якщо ви ще не зберігали налаштування, і хочете повернутися до первинних настройок, використовуйте iptables-restore:

В Ubuntu / Debian
sudo iptables-restore </etc/iptables.rules
В Centos / Redhat
iptables-restore </etc/sysconfig/iptables

Завдання до виконання

1. Ознайомитися з довідковою інформацією команди *iptables*.
2. Освоїти техніку роботи з трансляцією мережних адрес (NAT).
3. Збереження змін в *iptables.*

Контрольні питання

1. Які ланцюжки містяться в таблицях nat?
2. Опишіть роботу MASQUARADE при з'єднанні хоста з локальної мережі з хостом з зовнішньої мережі.
3. Опишіть дії ACCEPT, DROP, REJECT, RETURN, MARK, DELUDE. Чи всі дії перераховані? Якщо ні, то опишіть неперераховані дії.

Робота № 10. Функції та структура сервера імен. Сервер bind. Первинний (primary) сервер імен. Налаштування сервера імен. Вторинний (secondary) сервер імен. Кешуючий сервер імен.

***Мета роботи**:* вивчити поняття доменних імен, розібратися з призначенням і принципом функціонування служби доменних імен (DNS), ознайомитися з базовими поняттями протоколу DNS, навчитися встановлювати і налаштовувати найпростіший варіант DNS-сервера.

Короткі теоретичні відомості

BIND – найбільш поширений open-source додаток, в якому реалізовані протоколи DNS, що надають можливість перетворення доменних імен в IP-адреси і навпаки.

Зосередимося на прикладі налаштування своєї зони і файлу конфігурації для домену / вузла з підтримкою сервісів www і електронної пошти.

У нашому прикладі ми будемо використовувати наступні параметри:

IP-адреса, на якій буде встановлено сервер імен: *172.31.0.122*

ім'я домену / вузла: *itproffi.com*

авторитативні сервера імен для зони itproffi.com: *ns1.itproffi.com* (172.31.1.10) і *ns2. itproffi.com* (172.31.1.11)

служби www і електронної пошти для itproffi.com використовуватимуть адресу 172.31.1.10

Установка сервера bind

Установка bind дуже проста – потрібно скористатися менеджером пакетів. У Debian і Ubuntu виконайте наступну команду:

apt-get install bind9 dnsutils

У CentOS або Fedora:

yum install bind dnsutils

Пакет dnsutils необов'язковий для запуску сервера bind, але для тестування конфігурації ми будемо користуватися командою dig з цього пакета.

Створення файлу зони DNS

Подальші приклади будуть для Ubuntu/Debian, але також підходять і для Centos/RedHat, тільки директорія з настройками зон в CentOS буде знаходитися в /etc/named/, а основний файл конфігурації /etc/named.conf. Для початку нам потрібно створити новий файл зони для домену itproffi.com. Перейдіть в директорію /etc/bind/. Створіть в ній піддиректорію zones/master/ і перейдіть в неї, виконавши наступну послідовність команд:

cd /etc/bind

mkdir -p zones/master

cd zones/master/

Директорія */etc/bind/zones/master* буде містити файл зони для домену *itproffi.com*. При бажанні можна використовувати іншу директорію. Файл зони *db.itproffi.com* буде містити запис DNS, яка допоможе серверу імен встановити відповідність повного доменного імені IP-адресу. Створіть цей файл з наступним змістом:

```
  GNU nano 4.8                    /etc/bind/zones/master/db.itproffi.com
;
; Файл данных BIND для itproffi.com
;
$TTL 3h
@ IN SOA ns1.itproffi.com. admin.itproffi.com. (
1 ; серийный номер
3h ; обновление каждые 3 часа
1h ; повторная попытка через час
1w ; срок годности - 1 неделю
1h ) ; хранение кэша отказов 1 час;
@ IN NS ns1.itproffi.com.
@ IN NS ns2.itproffi.com.

itproffi.com. IN MX 10 mail.itproffi.com.
itproffi.com. IN A 172.31.1.10
ns1 IN A 172.31.1.10
ns2 IN A 172.31.1.11
www IN CNAME itproffi.com.
mail IN A 172.31.1.10
ftp IN CNAME itproffi.com.
```

Розглянемо ключові рядки цього файлу:

- Запис SOA: авторитативний сервер імен для itproffi.com – це ns1.itproffi.com, адреса відповідального за зону DNS адміністратора – admin@itproffi.com
- Записи NS: два сервера імен для зони itproffi.com – ns[1,2].itproffi.com

- Запис MX: поштовий сервер для itproffi.com. Число 10 означає рівень пріоритету
- Записи A: A означає «адрес» (address). Іншими словами, ns1 в зоні itproffi.com матиме адресу 172.31.1.10
- Запис CNAME (Canonical Name – канонічне ім'я): прив'язує одне доменне ім'я до іншого (канонічного), наприклад, встановлює відповідність mail.itproffi.com і itproffi.com.

Налаштування зворотної зони

На даному етапі DNS-сервер bind може видати IP-адресу, пов'язану з вузлом itproffi.com. Тепер нам потрібно навчити наш сервер імен зворотного процесу, тобто встановлювати відповідність імені IP-адресі. Для цього створимо ще один файл db.172.31.1 наступного змісту:

```
  GNU nano 4.8                    /etc/bind/zones/master/db.172.31.1
;
; Обратный файл даных BIND для 1.31.172.in-addr.arpa
;
$TTL 604800
1.31.172.in-addr.arpa. IN SOA ns1.itproffi.com. admin.itproffi.com. (
1 ; серийный номер
3h ; обновление каждые 3 часа
1h ; повторная попытка через час
1w ; срок годности - 1 неделя
1h ) ; хранение кэша отказа 1 час;
1.31.172.in-addr.arpa. IN NS ns1.itproffi.com.
1.31.172.in-addr.arpa. IN NS ns2.itproffi.com.
10.1.31.172.in-addr.arpa. IN PTR itproffi.com.
```

Запис PTR: DNS-запис, що використовується для визначення відповідності IP-адреси імені вузла.

Налаштування файлу конфігурації bind

На даний момент у нас повинні бути два файли:

/etc/bind/zones/master/db.itproffi.com

/etc/bind/zones/master/db.172.31.1

Тепер потрібно вставити імена обох файлів зони в файл конфігурації bind */etc/bind/named.conf.local*. Для цього додайте в файл наступні рядки:

zone «itproffi.com» {

type master;

file «/etc/bind/zones/master/db.itproffi.com»;

};

zone «1.31.172.in-addr.arpa» {
type master;
file «/etc/bind/zones/master/db.172.31.1»;
};

Останній момент перед перевіркою конфігурації – внести в файл *named.conf.options* IP-адрес стабільного DNS-сервера. Він застосовується у випадку, коли локальний DNS-сервер не буде знати відповідь на запит дозволу імені. Часто ця адресу надається інтернет-провайдером, але якщо ви шанувальник Google, можна вказати адресу 8.8.8.8 або 8.8.4.4.

Замініть наступний блок тексту в файлі *named.conf.options*:

// forwarders {
// 0.0.0.0;
//};

на блок тексту з адресою стабільного DNS-сервера

forwarders {
8.8.4.4;
};

Якщо ви плануєте, що до вашого сервера будуть підключатися інші комп'ютери, то потрібно дозволити в опціях зовнішні підключення. Для цього в основному файлі конфігурації, в секції options додайте або замініть наступні правила:

listen-on port 53 {any; };
allow-query {any; };

А краще, для безпеки, замість *any* пропишіть ваші мережі, з яких дозволено підключення:

listen-on port 53 {192.168.0.0/24; };
allow-query {192.168.0.0/24; };

Якщо цього не зробити, то при спробі звернення до сервера з іншого комп'ютера ви отримаєте помилку

Query refused

Перевірка файлів зони і конфігурації

Перш ніж спробувати запустити сервер імен з новою зоною і конфігурацією, можна скористатися деякими інструментами, щоб перевірити, що конфігурація коректна і не містить помилок.

Для перевірки файлів конфігурації виконайте наступну команду:

named-checkconf

З цією командою працює просте правило: відсутність результату – це хороший результат. Якщо команда нічого не повертає, значить помилок у ваших файлах конфігурації не виявлено.

Для перевірки файлів зони DNS можна скористатися командою named-checkzone:

named-checkzone itproffi.com /etc/bind/zones/master/db.itproffi.com

Перевірка зворотної зони

named-checkzone 1.31.172.in-addr.arpa /etc/bind/zones/master/db.172.31.1

Запуск і перезапуск сервера bind

Тепер ми можемо запускати сервер bind:

/etc/init.d/bind9 start

Якщо сервер уже був запущений, його можна перезапустити командою restart:

/etc/init.d/bind9 restart

Для того щоб перечитати конфігурацію не перезапускаючи сервер, використовуйте команду

rndc reload

Примітка! В Ubuntu 20.04 LTS BIND9 запускається як служба, а не як сервіс. Тому команда service bind9 start | stop | restart | status в даному дистрибутиві не діє. За замовчуванням після установки BIND9 запускається автоматично. Ви можете перевірити його статус за допомогою команди:

sudo systemctl status bind

Завдання до виконання

1. Ознайомитися з встановленням і настроюванням DNS сервера BIND в Linux.
2. Провести настроювання і тестування DNS-сервера.

Контрольні питання

1. Що таке DNS сервер?
2. Способи налаштування динамічного оновлення зон DNS-сервером.
3. Які ви знаєте зони DNS сервера.

Робота № 11. Функції та структура web-сервера. Сервер Apache. Налаштування apache

Мета роботи: ознайомитися з налаштуванням веб-сервера на операційній системі Linux.

Короткі теоретичні відомості

HTTP-сервер Apache – найбільш використовуваний веб-сервер у світі. Він має безліч потужних функцій, включаючи динамічно завантажувані модулі, надійну підтримку різних медіаформатів і інтеграцію з іншим популярним програмним забезпеченням.

Установка Apache

Apache доступний у репозиторіях програмного забезпечення Ubuntu за замовчуванням, і тому його можна встановити за допомогою стандартних інструментів управління пакетами.

Для початку вивантажимо покажчик локальних пакетів, щоб відобразити останні зміни на попередніх рівнях:

sudo apt update

Потім встановимо пакет apache2:

sudo apt install apache2

Після підтвердження установки *apt* встановить Apache і всі необхідні залежності.

Налаштування брандмауера

Перш ніж тестувати Apache, необхідно змінити налаштування брандмауера, щоб дозволити доступ до веб-портів за замовчуванням.

Під час установки Apache реєструється в UFW, надаючи кілька профілів додатків, які можна використовувати для включення або відключення доступу до Apache через брандмауер.

Виведіть список профілів додатків *ufw*, ввівши таку команду:

sudo ufw app list

Ви побачите список профілів додатків:

```
root@ubuntuSRV:~# ufw app list
Доступные приложения:
  Apache
  Apache Full
  Apache Secure
  Bind9
  CUPS
  OpenSSH
  Postfix
  Postfix SMTPS
  Postfix Submission
root@ubuntuSRV:~# _
```

Як показав результат, є три профілі, доступних для Apache:

- Apache: цей профіль відкриває тільки порт 80 (нормальний веб-трафік без шифрування)
- Apache Full: цей профіль відкриває порт 80 (нормальний веб-трафік без шифрування) і порт 443 (трафік з шифруванням TLS / SSL)
- Apache Secure: цей профіль відкриває тільки порт 443 (трафік з шифруванням TLS / SSL)

Рекомендується застосовувати самий обмежений профіль, який буде дозволяти заданий трафік. Оскільки в цьому модулі ми ще не налаштували SSL для нашого сервера, нам потрібно буде тільки дозволити трафік на порту 80:

sudo ufw allow 'Apache'

Перевірка веб-сервера

В кінці процесу установки Ubuntu запускає Apache. Веб-сервер уже повинен бути запущений і працювати.

Використовуйте команду ініціалізації systemd, щоб перевірити роботу служби:

sudo systemctl status apache2

```
root@ubuntuSRV:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2021-02-28 17:04:55 CET; 2h 11min ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 811 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 858 (apache2)
      Tasks: 7 (limit: 4614)
     Memory: 17.0M
     CGroup: /system.slice/apache2.service
             ├─ 858 /usr/sbin/apache2 -k start
             ├─ 891 /usr/sbin/apache2 -k start
             ├─ 892 /usr/sbin/apache2 -k start
             ├─ 893 /usr/sbin/apache2 -k start
             ├─ 894 /usr/sbin/apache2 -k start
             ├─ 895 /usr/sbin/apache2 -k start
             └─3189 /usr/sbin/apache2 -k start
```

Висновок підтвердив, що служба успішно запущена. Однак найкраще протестувати її запуск за допомогою запиту сторінки з Apache.

Відкрийте сторінку Apache за замовчуванням, щоб підтвердити роботу програмного забезпечення через ваш IP-адрес. Якщо ви не знаєте IP-адресу вашого сервера, є кілька способів дізнатися його за допомогою командного рядка.

Спробуйте ввести в командному рядку сервера наступну команду:

hostname -I

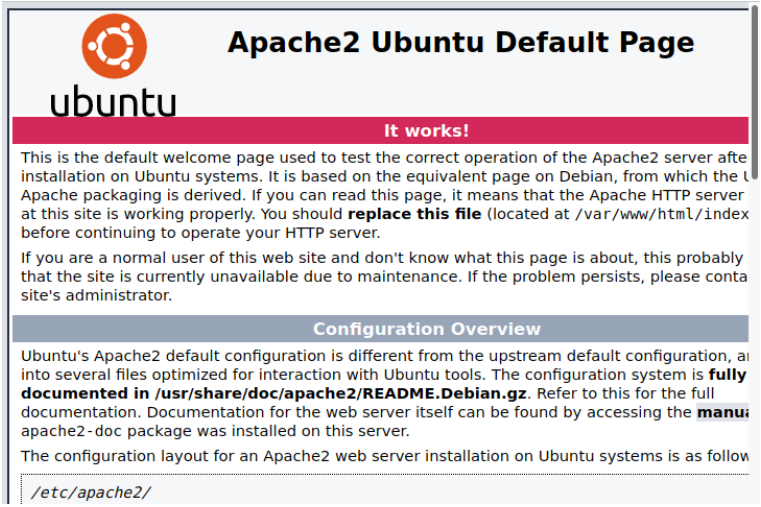
Ще один варіант – використовувати інструмент icanhazip, який повинен видати вам ваш публічний IP-адрес, видимий в іншому розташуванні в Інтернеті:

curl -4 icanhazip.com

Коли ви дізнаєтеся IP-адресу вашого сервера, введіть її в адресний рядок браузера:

http:/ your_server_ip

Ви побачите веб-сторінку Ubuntu Apache за замовчуванням:



Ця сторінка показує, що Apache працює коректно. Також на ній міститься інформація про важливі файли Apache і розташування каталогів.

Управління процесом Apache

Щоб зупинити веб-сервер, введіть:

sudo systemctl stop apache2

Щоб запустити зупинений веб-сервер, введіть:

sudo systemctl start apache2

Щоб зупинити і знову запустити службу, введіть:

sudo systemctl restart apache2

Якщо ви просто вносите зміни в конфігурацію, в багатьох випадках Apache може перезавантажуватися без відключення з'єднань. Для цього потрібно використовувати наступну команду:

sudo systemctl reload apache2

За замовчуванням Apache налаштований на автоматичний запуск при завантаженні сервера. Якщо ви не хочете цього, відключіть за допомогою наступної команди:

sudo systemctl disable apache2

Щоб перезавантажити службу для запуску під час завантаження, введіть:

sudo systemctl enable apache2

Тепер Apache повинен запуститися автоматично при наступному завантаженні сервера.

Налаштування віртуальних хостів

При використанні веб-сервера Apache ви можете використовувати віртуальні хости (аналогічні серверним блокам в Nginx) для інкапсуляції даних конфігурації та розміщення на одному сервері декількох доменів. Ми створимо домен *your_domain*, але ви повинні замінити це ім'я власним доменним ім'ям.

В Apache в Ubuntu 20.04 за замовчуванням включений один серверний блок, налаштований на обслуговування документів з директорії /var/www/html. Хоча це добре працює для окремого сайту, при хостингу декількох сайтів це незручно. Замість зміни /var/www/html ми створимо в /var/www структуру директорій для нашого сайту your_domain, залишивши /var/www/html як директорію за замовчуванням для виведення в разі, якщо запиту клієнта не відповідають ніякі інші сайти.

Створіть наступну директорію для your_domain:

sudo mkdir /var/www/your_domain

Потім призначте володіння Директорією за допомогою змінної середовища $USER:

sudo chown -R $USER:$USER /var/www/your_domain

Дозволи кореневих директорій веб-сервера повинні бути правильними, якщо ви не змінювали значення *umask*, яке встановлює дозволи файлу за замовчуванням. Щоб переконатися, що дозволи коректні, дозволити власнику читати, писати і запускати файли, а групам і іншим користувачам дозволити тільки читати і запускати файли, ви можете ввести наступну команду:

sudo chmod -R 755 /var/www/your_domain

Потім створіть як прикладу сторінку index.html, використовуючи *nano* або свій улюблений редактор:

sudo nano /var/www/your_domain/index.html

Додайте в сторінку наступний зразок коду HTML:

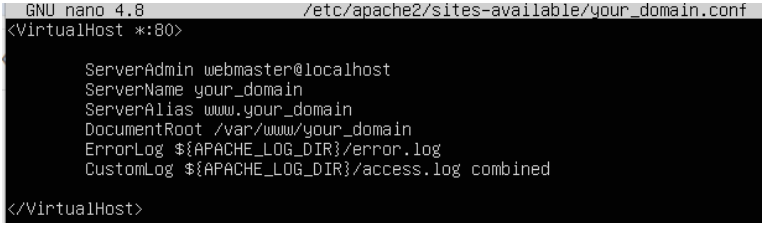
Збережіть файл і закрийте його після завершення.

Для обслуговування цього контенту Apache необхідно створити файл віртуального хоста з правильними директивами.

Замість зміни файлу конфігурації за замовчуванням, розташованого в */etc/apache2/sites-available/000-default.conf*, ми створимо новий файл в */etc/apache2/sites-available/your_domain.conf*:

sudo nano /etc/apache2/sites-available/your_domain.conf

Введіть наступний блок конфігурації, який схожий на заданий за замовчуванням, але оновлений з урахуванням нової директорії і доменного імені:



Зверніть увагу, що ми змінили *DocumentRoot* на нову директорію, а *ServerAdmin* – на адресу електронної пошти, доступний адміністратору сайту *your_domain*. Також ми додали дві директиви: директиву *ServerName*, яка встановлює базовий домен і повинна відповідати визначенню віртуального хоста, і директиву *ServerAlias*, яка задає додаткові імена, які повинні давати збіг, якщо б вони були базовими іменами.

Збережіть файл і закрийте його після завершення.

Активуємо файл за допомогою інструменту a2ensite:

sudo a2ensite your_domain.conf

Вимкніть сайт за замовчуванням, визначеним у 000-default.conf:

sudo a2dissite 000-default.conf

Потім перевіримо помилки конфігурації:

sudo apache2ctl configtest

Перезапустимо Apache для внесення змін:

sudo systemctl restart apache2

Тепер Apache повинен обслуговувати ваше доменне ім'я.

Основні файли і директорії Apache

Контент

- /var/www/html: веб-контент, до складу якого за замовчуванням входить тільки показана раніше сторінка Apache за замовчуванням, виводиться з каталогу /var/www/html. Це можна змінити шляхом зміни файлів конфігурації Apache.

Конфігурація сервера

- /etc/apache2: каталог конфігурації Apache. Тут зберігаються всі файли конфігурації Apache.
- /etc/apache2/apache2conf: головний файл конфігурації Apache. Його можна змінити для внесення змін у глобальну

конфігурацію Apache. Цей файл відповідає за завантаження багатьох інших файлів у каталозі конфігурації.

- /etc/apache2/ports.conf: цей файл задає порти, які буде прослуховувати Apache. За замовчуванням Apache прослуховує порт 80, а якщо активований модуль з функціями SSL, він також прослуховує порт 443.
- /etc/apache2/sites-available/: каталог, де можна зберігати віртуальні хости для кожного сайту. Apache не використовуватиме файли конфігурації з цього каталогу, якщо вони не будуть пов'язані з каталогом sites-enabled. Зазвичай всі зміни конфігурації серверних блоків виконуються в цьому каталозі, а потім активуються за допомогою посилання на інший каталог за допомогою команди a2ensite.
- /etc/apache2/sites-enabled/: каталог, де зберігаються активні віртуальні хости для кожного сайту. Зазвичай вони створюються за допомогою створення посилань на файли конфігурації з каталогу sites-available за допомогою команди a2ensite. Apache зчитує файли конфігурації і посилання з цього каталогу при запуску або перезавантаженні, коли компілюється повна конфігурація.
- /etc/apache2/conf-available/, /etc/apache2/conf-enabled/: ці каталоги мають ті ж відносини, що і каталоги sites-available і sites-enabled, але використовуються для зберігання фрагментів конфігурації, які не належать віртуальному хосту. Файли з каталогу conf-available можна активувати за допомогою команди a2enconf і відключити за допомогою команди a2disconf.
- /etc/apache2/mods-available/, /etc/apache2/mods-enabled/: ці каталоги містять доступні і активовані модулі відповідно. Файли з розширенням .load містять фрагменти для завантаження певних модулів, а файли з розширенням .conf містять конфігурації цих модулів. Модулі можна активувати і відключати за допомогою команд a2enmod і a2dismod.

Журнали сервера

- /var/log/apache2/access.log: за замовчуванням кожен запит веб-сервера реєструється в цьому файлі журналу, якщо Apache не налаштований по-іншому.

- /var/log/apache2/error.log: за замовчуванням всі помилки реєструються в цьому файлі. Директива LogLevel у конфігурації Apache вказує, наскільки детальні записи реєструються в журналах помилок.

Завдання до виконання

1. Опишіть основні функціональні можливості Apache.
2. Дайте порівняльну характеристику між класичним способом установки веб сервера і використанням комплексу.
3. Налаштування віртуальних хостів.

Контрольні питання

1. За якими параметрами оцінюється веб-сервер?
2. Перерахуйте найбільш поширені веб-сервера. Охарактеризуйте коротко кожен.
3. У яких випадках організація локального веб-сервера може бути необхідною?
4. Як змусити Apache працювати на порту, що відрізняється від прийнятого за замовчуванням?

Робота № 12. Функції та структура проксі-сервера. Сервер squid. Налаштування squid

Мета роботи: ознайомитися з налаштуванням проксі-сервера *Squid* на операційній системі Linux.

Короткі теоретичні відомості

Squid – найпопулярніший HTTP-проксі сервер для кешування і перенаправлення. Він широко використовується різними компаніями для кешування веб-сторінок з веб-сервера для підвищення швидкості роботи, зниження часу відповіді та обмеження використання пропускної здатності мережі. Розглянемо установку проксі-сервера *squid* і його використання як HTTP-проксі сервера.

Установка проксі-сервера Squid

Перш ніж почати, варто відзначити, що сервер Squid не вимагає значних ресурсів, але використання оперативної пам'яті може змінюватися в залежності від кількості клієнтів, які здійснюють доступ в інтернет через проксі-сервер.

Пакет Squid доступний у стандартному репозиторії

В Ubuntu/Debian

sudo apt-get -y install squid

В Redhat/Centos

yum install -y squid

Запустіть його і задайте запуск при завантаженні:

sudo systemctl start squid

sudo systemctl enable squid

Після цього можна перевірити статус служби:

sudo systemctl status squid

Важливі файли Squid розташовуються в наступних директоріях:

- файл конфігурації: /etc/squid/squid.conf
- журнал доступу: /var/log/squid/access.log
- журнал кешу: /var/log/squid/cache.log

Файл конфігурації за замовчуванням містить ряд директив, за допомогою яких здійснюється управління роботою

сервера. Для внесення змін відкрийте файл будь-яким текстовим редактором:

sudo vim /etc/squid/squid.conf

У ньому досить багато параметрів, ми розглянемо найважливіші з них:

http_port – порт HTTP-проксі сервера, за замовчуванням 3128. Для безпеки рекомендується змінити його на інший.

visible_hostname – параметр використовується для визначення імені вузла сервера *Squid*. Можна задати будь-яке ім'я.

Також можна вказати параметр *intercept* (або *transparent* для старих версій), наприклад, *http_port 3128 intercept*. У цьому випадку ваш сервер буде працювати як прозорий проксі (без необхідності налаштовувати його використання на стороні клієнта).

Для подальшого розуміння роботи проксі потрібно зрозуміти такі параметри:

http_access – даний параметр регулює доступ до HTTP-проксі сервера. За допомогою нього можна дозволити або заборонити доступ через сервер як до певних ресурсів в інтернеті, так і до певних груп користувачів.

В даний момент будь-який доступ заборонений (*deny all*). Щоб почати використання сервера, потрібно змінити його, наприклад, на *http_access allow all* (дозволити будь-який доступ). Параметр all можна замінити на назву списку доступу, який ми розглянемо трохи нижче.

acl (access control list) – у цьому параметрі вказуються ресурси в інтернеті, порти, ір-адреси користувачів, локальні мережі. Загалом це список, до якого будуть застосовуватися різні правила. Таких списків може бути необмежена кількість.

Після внесення змін потрібно перезапустити *Squid* наступною командою:

sudo systemctl restart squid

Налаштування Squid як HTTP-проксі

Загальний синтаксис у проксі виглядає наступним чином:

http_access [allow / deny] [назва acl]

При описі контролю доступу можна використовувати оператор заперечення «!». Наприклад, наступний рядок забороняє доступ до всіх портів, крім описаних у листі *Safe_ports*

http_access deny !Safe_ports

Додавання списків контролю доступу

Розглянемо створення списків доступу *acl* докладніше. За замовчуванням вже є попередньо налаштовані *acl localnet*

```
acl localnet src 0.0.0.1-0.255.255.255  # RFC 1122 "this" network (LAN)
acl localnet src 10.0.0.0/8             # RFC 1918 local private network (LAN)
acl localnet src 100.64.0.0/10          # RFC 6598 shared address space (CGN)
acl localnet src 169.254.0.0/16         # RFC 3927 link-local (directly plugged) machines
acl localnet src 172.16.0.0/12          # RFC 1918 local private network (LAN)
acl localnet src 192.168.0.0/16         # RFC 1918 local private network (LAN)
acl localnet src fc00::/7               # RFC 4193 local private network range
acl localnet src fe80::/10              # RFC 4291 link-local (directly plugged) machines
```

Ви можете його відредагувати або видалити. Створимо новий *acl*

sudo vim/etc/squid/squid.conf

Додайте правило такого вигляду:

acl boss src XX.XX.XX.XX

де *boss* – ім'я списку контролю доступу, *src* – параметр, що задає адресу джерела(source), а *XX.XX.XX.XX* – IP-адреса машини клієнта (можна також вказувати підмережі або діапазони). Нові списки контролю доступу потрібно додавати до початку розділу *ACL*. Аналогічним чином можна створювати списки доступу з обмеженням за адресою місця призначення (параметр *dst* замість *src*), а також використовувати замість адрес доменні імена (*srcdomain* для джерела, *dstdomain* для місця призначення).

Дуже бажано поруч з *ACL* вказувати коментар з коротким описом користувача цієї IP-адреси, наприклад:

acl boss src 192.168.0.102 # IP-адреса керівника

Після цього потрібно дозволити доступ для *boss*:

http_access allow boss

Щоб зміни вступили в силу, потрібно перезавантажити Squid:

sudo systemctl restart squid

Відкриття портів

За замовчуванням в конфігурації Squid дозволено використання тільки певних портів.

```
acl SSL_ports port 443
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # unregistered ports
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
```

Якщо потрібне використання додаткових портів, можна задати їх у файлі конфігурації:

acl Safe_ports port XXX

де *XXX* – номер порту, використання якого потрібно дозволити. Знову бажано пояснювати ACL коментарем. Не забуваємо перезапустити Squid для застосування налаштувань.

Робота проксі в прозорому режимі

Прозорий режим передбачає автоматичну роботу проксі-сервера без необхідності в явному вигляді вказувати його на клієнтських машинах. У загальному випадку клієнт може взагалі не знати, що працює через проксі. Це може бути корисним для забезпечення анонімності, обмеження доступу до деяких сайтів і навіть економії мережевого трафіку, оскільки проксі-сервер може стискати дані.

Крім вже розглянутої вище опції *intercept* в параметрі *http_port* файлу конфігурації, для забезпечення належного функціонування прозорого проксі потрібно відповідним чином налаштувати маршрутизатор, щоб всі вхідні і вихідні запити на порт 80 перенаправлялись на порт, який використовується проксі-сервером.

У разі використання *iptables* потрібно додати наступні правила (в розглянутому прикладі *eth1* – внутрішній інтерфейс, *eth0* – зовнішній, *SQUID_IP* - IP-адресу проксі-сервера):

iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j DNAT --to SQUID_IP:3128

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3128

Налаштування параметрів кешування

Одна з важливих функцій проксі-сервера – кешування вебсторінок для розвантаження веб-сервера і прискорення доступу. Squid підтримує два типи кешу, в оперативній пам'яті і на жорсткому диску.

Кеш в оперативній пам'яті налаштовується наступними параметрами:

cache_mem 1024 MB – виділений для кешування обсяг пам'яті

maximum_object_size_in_memory 512 KB – максимальний розмір об'єкта в кеші

Параметри кеша на жорсткому диску задаються наступною директивою:

cache_dir тип_сховища шлях_до_сховища розмір ур_1 ур_2

Розмір вказується в мегабайтах, ур_1 і ур_2 – кількість директорій першого і другого рівня, відповідно, наприклад:

cache_dir ufs /var/squid_cache 2048 32 512

Аналогічно кешу в пам'яті за допомогою наступного параметра вказується максимальний розмір об'єкта в кеші на диску:

maximum_object_size 8 MB

Обмеження швидкості

Squid може обмежувати швидкість доступу до мережі. Хоча в сучасних умовах ця функція може здатися надмірною, вона часто може виявитися корисною, наприклад, для обмеження використання пропускної здатності каналу будь-якими автоматизованими завданнями.

Для реалізації обмеження швидкості Squid використовує механізм пулів затримки (delay pools). Пули затримки можна умовно уявити у вигляді ємності, яка «заповнюється» даними, і після цього «випускає» їх тільки з певною швидкістю. Кількість пулів задається у файлі конфігурації наступним чином:

delay_pools кількість_пулов

Кожен пул має номер (від 1 до заданої кількості), а також клас. Класи реалізують багатоступеневу структуру обмеження:

1 клас – загальне обмеження

2 клас – загальне обмеження і обмеження для підмереж

3 клас – загальне обмеження, обмеження для підмереж і обмеження для окремих ір-адрес

Класи пулів задаються директивою *delay_class*, в якості аргументів якої передаються номер пулу і клас:

delay_pools 2
delay_class 1 1
delay_class 2 2

Параметри пулів задаються директивою *delay_parameters* і описують максимальний обсяг пулу й обмеження на кожен рівень (в байтах) у залежності від класу. Наприклад, параметри для пулу 1 класу з номером 1:

delay_parameters 1 64000/256000

означатимуть, що після отримання перших 256 Кб запиту на максимальній швидкості швидкість буде обмежена 64 Кб/с, тобто 512 Кбіт/с.

Для 2 класу і вище аналогічним чином задаються обмеження для підмережі, окремої адреси і т. д., наприклад наступна директива обмежує загальну швидкість до 8 Мбіт/с, а швидкість для підмережі після перших 256 Кб запиту – до 512 Кбіт/с.

delay_parameters 2 512000/512000 64000/256000

Щоб задати пули затримки для певних списків контролю доступу, використовується директива *delay_access*, що містить номер пулу, параметр *allow* або *deny* і назву списку, наприклад:

delay_access 2 allow localnet

Блокування веб-сайтів

Для блокування доступу до небажаних веб-сайтів спочатку створіть файл з «чорним списком»:

touch /etc/squid/blacklisted_sites.acl

Тепер в цей файл потрібно додати сайти, до яких потрібно заблокувати доступ. Наприклад, заблокуємо доступ до однокласників і вконтакте:

.vk.com
.ok.ru

Точки перед іменами вказують *squid* блокувати всі посилання на ці сайти, в тому числі www.vk.com, subsite.vk.com і т. д.

Далі потрібно відкрити файл конфігурації

vim /etc/squid/squid.conf

і додати список контролю доступу по доменних іменах, вказаних у файлі, а також правило, що забороняє доступ для цього списку:

acl bad_urls dstdomain «/etc/squid/blacklisted_sites.acl»

http_access deny bad_urls

```
auth_param basic program /usr/lib64/squid/basic_ncsa_auth /etc/squid/passwd
auth_param basic children 5
auth_param basic realm Squid Basic Authentication
auth_param basic credentialsttl 2 hours
acl auth_users proxy_auth REQUIRED
acl bad_urls dstdomain "/etc/squid/blacklisted_sites.acl"
http_access deny bad_urls
http_access allow auth_users
```

Зверніть увагу на порядок розташування правил, правила доступу виконуються зверху вниз. Тому правило, що забороняє, розташоване вище правила, яке дозволяє.

Блокування по масці

Можна здійснювати блокування не тільки по іменах сайтів, але і по масці. Тобто заблокувати доступ до сайтів, в яких є певне поєднання букв. Аналогічним чином створюється файл зі списком заборонених ключових слів:

*touch /etc/squid/blockkeywords.l*st

Далі в нього додаються ключові слова, наприклад:

facebook

instagram

gmail

Відкрийте файл конфігурації і додайте до нього такі – список контролю доступу і правило:

acl blockkeywordlist url_regex «/etc/squid/blockkeywords.lst»

http_access deny blockkeywordlist

а потім збережіть файл і перезавантажте *squid*:

systemctl restart squid

```
auth_param basic program /usr/lib64/squid/basic_ncsa_auth /etc/squid/passwd
auth_param basic children 5
auth_param basic realm Squid Basic Authentication
auth_param basic credentialsttl 2 hours
acl auth_users proxy_auth REQUIRED
acl bad_urls dstdomain "/etc/squid/blacklisted_sites.acl"
acl blockkeywordlist url_regex "/etc/squid/blockkeywords.lst"
http_access deny bad_urls
http_access deny blockkeywordlist
http_access allow auth_users
```

Тепер усі сайти, в назві доменів яких зустрічаються facebook, instagram, gmail, будуть заблоковані.

Завдання до виконання

1. Встановіть та налаштуйте HTTP proxy сервер Squid під операційною системою Linux.
2. Опишіть базову конфігурацію Squid для роботи в режимі прозорого проксі.

Контрольні питання

1. Що таке списки контролю доступу в Squid? Перерахуйте основні критерії, що використовуються при формуванні списків контролю доступу в Squid.
2. Перерахуйте види ідентифікації в Squid. Назвіть методи ідентифікації за логіном / паролем, підтримувані в Squid?
3. Наведіть приклад конфігурації Squid, що забороняє скачування великих файлів.

Робота № 13. Налаштування proftpd. Функції та структура ssh-сервера. Сервер sshd

Мета роботи: ознайомитися з налаштуванням FTP-сервера на прикладі vsFTPd на операційній системі Linux.

Короткі теоретичні відомості

File Transfer Protocol, тобто FTP – протокол передачі файлів, призначений для передачі файлів між віддаленими комп'ютерами через мережу. Незважаючи на те, що сам протокол FTP є на сьогоднішній день не найдосконалішим через те, що дані, які передаються, не шифруються, однак це не робить його застарілим. Крім того, все-таки до FTP можливо застосовувати криптографічний захист на основі протоколу SSL, що і робить FTP гідним інструментом для передачі файлів.

Оскільки FTP працює за схемою клієнт-серверної взаємодії, то вміла і надійна реалізація протоколу (та й взагалі системи) в змозі забезпечити йому надійний захист, високу швидкість і, як наслідок – популярність, що і можна спостерігати на сьогоднішній день, адже більшість великих проєктів, таких як ftp.gnu.org, ftp.suse.com, ftp.redhat.com, ftp.gnome.org і т. д., використовують для поширення програмного забезпечення зі своїх серверів саме FTP. Треба зауважити, що такою популярністю FTP зобов'язаний, у більшій мірі, однією зі своїх численних реалізацій-vsFTPd. Це FTP-сервер, що підтримує роботу з найсучаснішими технологіями із захисту даних – SSL і IPv6, його реалізація поєднує в собі високу надійність, стабільність, швидкість роботи і передачі даних, а також гнучке налаштування роботи сервера і широкий функціонал. Розробником vsFTPd є Кріс Еванс – професійний дослідник у сферах захисту даних та інформаційної безпеки. vsFTPd є FTP-сервером за замовчуванням практично у всіх Linux-системах, оскільки, як уже згадувалося, крім надійності і високою швидкості роботи, володіє великими функціональними можливостями, найбільш значущими з яких є:

- робота з віртуальними користувачами;
- робота з віртуальними IP-адресами;

- конфігурація користувачів;
- підтримка;
- SSL-шифрування для захисту переданих даних;
- контроль смуги пропускання.

Установка vsFTPd

Встановити *vsftpd* у Debian-орієнтованих дистрибутивах Linux дозволяє команда:

apt-get install vsftpd

Також корисно буде встановити і простий FTP-клієнт для тестування з'єднання і передачі файлів:

apt-get install ftp

Для дистрибутивів на основі RPM-пакетів, CentOS, RedHat:

yum install vsftpd

yum install ftp

Примітка: *yum* – це пакетний менеджер, той же самий *apt*, але адаптований для управління пакетами формату * *.rpm*.

Після установки для vsFTPd буде доступна технічна документація, яка зазвичай знаходиться в каталозі /usr/share/doc/vsftpd/examples – тут наведені варіанти різних конфігурацій, в залежності від характеру і масштабів використання vsFTPd. Ознайомитися з документацією можна також за допомогою команди:

man vsftpd.conf

Зазвичай після установки демон vsFTPd автоматично запускається, переконатися в цьому можна за допомогою наступних команд:

systemctl status vsftpd

або:

service vsftpd status

Запуск, перезапуск і зупинка сервера:

systemctl start vsftpd

systemctl restart vsftpd

systemctl stop vsftpd

Для включення демона *vsftpd* в автозавантаження використовується команда:

systemctl enable vsftpd

Якщо vsFTPd використовується в серверних дистрибутивах, в яких часто працює фаєрвол, наприклад *ufw*, то ще може знадобитися дозволити використання портів 20 і 21:

ufw allow 20 /tcp

ufw allow 21 /tcp

Налаштування vsFTPd

Конфігураційним файлом для настройки *vsFTPd* є файл *vsftpd.conf*, який зазвичай знаходиться в каталозі *etc/*. Ознайомитися з його складом cat можна командою *cat*:

cat /etc/vsftpd.conf

Про всяк випадок корисно перед редагуванням оригінального файлу налаштувань зробити його резервну копію:

cp /etc/vsftpd.conf vsftpd.conf.backup

FTP-сервер vsFTPd передбачає два основні варіанти роботи: з анонімними й авторизованими користувачами. Перший варіант вважається «безпечнішим», але тільки тому, що для забезпечення надійного захисту практично нічого налаштовувати і не потрібно. Але при грамотній організації авторизованого доступу, який передбачає роботу з FTP локальних користувачів системи, можна забезпечити безпеку нітрохи не гірше, ніж при використанні анонімного доступу.

Налаштування в режимі анонімного доступу

Робота vsFTPd у даному режимі полягає в тому, що дії з файлами на віддаленому сервері виконуються одним певним за замовчуванням користувачем, наприклад, користувачем з ім'ям «ftp» або «anonymous», при цьому як пароль використовується e-mail.

Щоб включити анонімний доступ по FTP, потрібно в файлі vsftpd.conf визначити значення «YES» для відповідної директиви:

anonymous_enable = YES

Тепер для управління файлами буде використовуватися певний каталог (зазвичай це /srv/ftp) і певний користувач – зазвичай ftp.

Можна визначити й інше розташування файлів для анонімного доступу по FTP, тобто змінити домашню директорію користувача ftp:

mkdir /srv/share/ftp

usermod -d /srv/share/ftp ftp

Якщо потрібно, щоб анонімні користувачі могли ще й завантажувати файли на віддалений сервер, то це дозволить зробити директива:

anon_upload_enable = YES

Тепер можна скопіювати необхідні для анонімного доступу файли в домашню папку користувача ftp і перезапустити демон vsftpd:

systemctl restart vsftpd

Зазвичай цього набору налаштувань досить для організації анонімного FTP-доступу. Для перевірки з'єднання можна виконати команду ftp address_host:

ftp 127.0.0.1

або:

ftp localhost

Налаштування в режимі авторизованого доступу

Для авторизованого доступу найпростіше включити використання локальних облікових записів на сервері. Для цього потрібно вказати наступну директиву в файлі конфігурації vsftpd.conf:

local_enable = YES

Для завдання дозволу локальним авторизованим користувачам завантажувати файли на сервер вказується директива:

write_enable = YES

Що в разі успішного виконання дасть приблизно такий висновок:

Тепер потрібно перезапустити vsftpd для активації зроблених змін:

systemctl restart vsftpd

Обмеження користувачів у своїх домашніх каталогах

Для визначення користувачів, яким дозволений доступ тільки до свого домашнього каталогу, існують директиви:

chroot_list_enable = YES

chroot_list_file = /etc/vsftpd.chroot_list

Перша включає використання списку користувачів, друга визначає файл, в якому в кожному рядку перераховані користувачі з доступом тільки до своїх домашніх каталогів. Якщо при цьому ще вказати директиву:

chroot_local_user = YES

яка «замикає» в *chroot()* локальних користувачів і вище своїх домашніх каталогів вони піднятися не зможуть, то в цьому випадку користувачі в файлі *vsftpd.chroot_list* обмежуватися своїми домашніми каталогами не будуть, на відміну від тих, хто в цей список не внесений.

При розподілі FTP-доступу серед користувачів може виникнути помилка, яка виникає через те, що локальний користувач має права на запис в корені домашнього каталогу, що з міркувань безпеки неприпустимо. Ця помилка виглядає наступним чином:

500 OOPS: vsftpd: refusing to run with writable root inside chroot ()

Кращим способом виправити цю помилку є вказівка якогось загального кореневого каталогу, куди при підключенні будуть потрапляти всі користувачі, які мають доступ лише до своїх домашніх піддиректорій, наприклад:

local_root = /home

Можна також для усунення цієї помилки відключити перевірку на запис в домашній каталог:

allow_writeable_chroot = YES

Але все ж це варто робити лише тоді, коли є чітке розуміння, навіщо це потрібно для конкретної ситуації.

Захист даних за допомогою SSL

Щоб мати можливість шифрувати передані дані, слід налаштувати vsFTPd в режимі FTPS. Це та ж сама передача даних по FTP, але організована поверх SSL – протоколу, який шифрує і перевіряє дані за допомогою сертифікатів і ключів.

Для включення режиму FTPS потрібно задіяти таку директиву:

ssl_enable = Yes

За замовчуванням у файлі конфігурації vsftpd.conf також присутні опції, що визначають сертифікати і ключі, наприклад:

rsa_cert_file = /etc/ssl/certs/ssl-cert-snakeoil.pem

rsa_private_key_file = /etc/ssl/private/ssl-cert-snakeoil.key

Ці сертифікат і ключ необхідно замінити. Для використання FTPS необхідно використовувати сертифікат і ключ, які згенеровано (або отримано) для конкретного сервера або комп'ютера.

Завдання до виконання

1. Встановіть та налаштуйте FTP сервер під операційною системою Linux.
2. Дозвольте анонімний доступ для всіх користувачів на даний ftp-сервер.
3. Перевірте роботу ftp-сервера з цією конфігурацією з вашої основної операційної системи.

Контрольні питання

1. Яке призначення ftp-сервера?
2. Яким чином проводиться настройка vsftpd?
3. Яке призначення мережевого протоколу SSL?

Робота № 14. Функції та структура поштового сервера. Сервер sendmail, postfix, mailq, exim. Налаштування postfix. Засоби боротьби зі спамом.

Мета роботи: на прикладі MTA (mail transfer agent) postfix вивчити конфігурацію поштового сервера в середовищі Linux.

Короткі теоретичні відомості

Postfix – це програмний продукт, що дозволяє організувати поштовий сервер. Він був створений як альтернатива Sendmail – найстаршому агенту передачі пошти (MTA – Mail Transfer Agent). Postfix поширюється з відкритим вихідним кодом і використовується розробниками для маршрутизації і пересилання поштових листів всередині системи Linux.

Вимоги

Для реалізації даного завдання нам буде потрібно:

- Сервер на Ubuntu 20.04, який буде функціонувати в режимі поштової служби. Потрібно, щоб він включав у себе користувача без прав *root* з привілеями *sudo*. Разом з цим також необхідний брандмауер, налаштований за допомогою *Uncomplicated Firewall (UFW)*.
- Доменне ім'я сервера. Якщо вам буде потрібно отримати доступ до пошти з зовнішнього джерела, то потрібно також подбати про MX-записи, яка буде вказувати на поштовий сервер.

Зверніть увагу на те, що настройка хоста буде виконуватися з доменним ім'ям типу *email.example2.co*m. В подальшому вам буде потрібно замінити всі подібні значення на своє власне доменне ім'я.

Як встановити Postfix

За умовчання Postfix включений у репозиторій операційної системи Ubuntu.

Насамперед оновлюємо кеш пакетів:

sudo apt update

Потім встановлюємо безпосередньо сам пакет Postfix. Тут важливо звернути увагу на значення DEBIAN_PRIORITY = low – воно дозволяє нам підключити деякі додаткові опції.

sudo DEBIAN_PRIORITY = low apt install postfix

В результаті перед нами відобразиться ряд питань і повідомлень. Відповідаємо на них наступним чином:

- *General type of mail configuration?* – в нашому випадку ми вказуємо значення «Internet Site», але воно може відрізнятися в залежності від інфраструктури.
- *System mail name* – стандартний домен, який використовується для створення точної адреси електронної пошти. Він необхідний в тих випадках, коли є тільки частина адреси з ім'ям профілю. Наприклад, ім'я сервера *email.example2.com*, але ми хочемо, щоб ім'я системної пошти виглядало як *example2.com*. В цьому випадку для користувача user буде використовуватися адреса *user@example2.com*.
- *Root and postmaster mail recipient* – це аккаунт Linux, на який буде перенаправлятися пошта, адресована *root@* і *postmaster@*. В даному випадку вам буде потрібно використовувати свій обліковий запис, наприклад, *UserNameZ7*.
- *Other destinations to accept mail for* – поточний параметр визначає одержувачів пошти, яких буде приймати цей екземпляр Postfix. При необхідності ви можете вказати власні доменні імена, для яких сервер буде отримувати пошту.
- *Force synchronous updates on mail queue?* – якщо у вашому випадку використовується журнальна файлова система, то вкажіть значення *«No»*.
- *Local networks* – це список локальних мереж, для яких поштова служба налаштована як пристрій передачі повідомлення. Початкові значення підійдуть для всіх, але якщо ви захочете їх змінити, то рекомендується максимально скоротити діапазон мереж.
- *Mailbox size limit* – дозволяє скоротити розмір повідомлення. Якщо встановлено значення «0», то в такому випадку обмеження будуть повністю відключені.

- *Local address extension character* – символ, який використовується для відділення звичайної частини адреси від розширення. За замовчуванням значення встановлено як «+», можете його залишити.
- *Internet protocols to use* – тут вказуємо, чи потрібно обмежувати версії IP, які підтримуються Postfix. Вибираємо значення *«All»*.

Всі вищевказані налаштування ми в будь-який момент часу можемо підкоригувати. Щоб відкрити вікно редагування, досить ввести:

sudo dpkg-reconfigure postfix

На цьому установка Postfix на Ubuntu завершена, тепер можемо переходити до більш детальних налаштувань.

Налаштування Postfix

Більшість налаштувань конфігурації Postfix задані у файлі *main.cf*, який можна знайти за адресою */etc/postfix/main.cf*. Тут ми можемо піти таким чином: змінювати параметри безпосередньо в самому файлі або скористатися командою *postconf*.

Для настройки Postfix насамперед нам потрібно прописати розташування пошти звичайного користувача Ubuntu. У нашому випадку ми використовуємо формат *Maildir* – в ньому повідомлення виділяються в окремі файли, що переміщуються між каталогами. Також ви можете зберігати повідомлення в форматі *mbox* або будь-якому зручному для вас.

Вказуємо значення *Maildir* для змінної *home_mailbox*. Налаштувати її можна за допомогою команди:

sudo postconf -e 'home_mailbox = Maildir/'

Прописуємо розташування таблиці *virtual_alias_maps*, де всі облікові записи пошти зіставляються з обліковими системи Linux. Також активуємо ще одну команду, за допомогою якої ми зіставимо розташування таблиці з файлом бази даних хеша в */etc/postfix/virtual*:

sudo postconf -e 'virtual_alias_maps = hash:/etc/postfix/virtual'

Тепер ми можемо почати зіставлення облікових записів пошти з профілями користувачів в ОС Linux. Для цього

створимо файл в nano, ви ж можете вибрати будь-який інший текстовий редактор:

sudo nano /etc/postfix/virtual

Записуємо всі адреси, для яких ми хочемо отримувати електронну пошту, а також прописуємо користувачів Ubuntu Postfix, яким будуть приходити листи. Для прикладу візьмемо таку ситуацію: нам потрібно отримувати всі листи на адреси *excontact@example2.com* і *exadmin@examlpe2.com* і відправляти їх користувачеві операційної системи Linux з ім'ям *UserNameZ7*. В такому випадку настройка файлу буде виглядати наступним чином:

excontact@example2.com UserNameZ7

exadmin@examlpe2.com UserNameZ7

Після успішного введення даних зберігаємося і виходимо з файлу. У нашому випадку це редактор nano, тому затискаємо на клавіатурі комбінацію клавіш «CTRL + X, Y» і тиснемо «Enter». Після цього здійснюємо зіставлення рядком коду:

sudo postmap /etc/postfix/virtual

Потім перезавантажуємося командою:

sudo systemctl restart postfix

Якщо брандмауер був налаштований за допомогою UFW, вам буде потрібно додати один виняток. Це пов'язано з тим, що UFW за замовчуванням блокує всі зовнішні підключення до служб сервера. Вирішити проблему можна одним рядком коду:

sudo ufw allow Postfix

Установка поштового клієнта

Встановимо пакет *s-nail* для взаємодії з поштою, що доставляється. Перед тим як почати установку, рекомендується перевірити настройку змінної середовища «*MAIL*». Клієнту ця змінна необхідна для того, щоб визначати місця пошти для користувача.

Якщо необхідно гарантовано задати змінну *MAIL*, незалежно від способу доступу до облікового запису, потрібно вказати її в файлі */etc/bash.bashrc* і додати в */etc/profile.d*, щоб вона використовувалася всіма користувачами.

Для цього введемо наступне:

echo 'export MAIL=~/Maildir' | sudo tee -a /etc/bash.bashrc | sudo tee -a /etc/profile.d/mail.sh

Щоб прочитати змінну поточного сеансу, можна ввести:

source /etc/profile.d/mail.sh

Тепер ми можемо переходити до установки клієнта *s-nail*. Прописуємо для цього:

sudo apt install s-nail

Поки що не запускаємо його, додамо в нього кілька записів. Спочатку відкриємо файл в редакторі nano:

sudo nano /etc/s-nail.rc

В кінець вставимо наступне:

. . .

set emptystart

set folder = Maildir

set record = + sent

Розшифруємо кожен рядок:

- *set emptystart* – дає можливість клієнту відкриватися навіть при порожній поштовій скриньці;
- *set folder = Maildir* – прописує для директорії Maildir змінну *folder*;
- *set record = + sent* – створює файл *sent* у форматі mbox для зберігання відправленої пошти в каталозі, заданому в змінній *folder*.

На цьому з файлом закінчуємо – збережемо та закриємо його. Тепер нам потрібно виконати ще одну дію – форматувати структуру *Maildir*. Щоб це зробити, необхідно відправити собі електронного листа командою *s-nail*. Оскільки файл *sent* доступний тільки після створення *Maildir*, потрібно відключити запис в цей файл. Для цього скористаємося командою *-Snorecord*.

Для відправлення листа додамо рядок в команду s-nail. Зверніть увагу, що в кінці вказується ім'я користувача – вам буде потрібно змінити його на своє.

echo 'init' | s-nail -s 'init' -Snorecord UserNameZ7

Переконаємося, що каталог був створений:

ls -R ~ / Maildir

В результаті має відобразитися приблизно наступне:

/home/UserNameZ7/Maildir /:

cur new tmp

/home/UserNameZ7/Maildir/cur:

/home /UserNameZ7/Maildir/new:

1463177269.Vfd01I40e4dM691221.mail.example.com

/home/UserNameZ7/Maildir/tmp:

Тепер ми можемо перейти до тестування поштового клієнта.

Тестування відправки пошти

Для початку запустимо клієнт. Для цього використовуємо команду:

s-nail

Щоб відправити повідомлення, передамо вміст текстового файлу в процес *s-nail*. Відкриємо для цього текстовий редактор:

nano ~/test_message

Пропишемо туди повідомлення, наприклад:

Вітання! Це тестове повідомлення.

Зберігаємося і закриваємо редактор. Для передачі повідомлення в *s-nail* використовуємо команду *cat*. Вона може набувати таких значень:

-s – задає рядок теми повідомлення;

-r – змінює поле «From», в якому за замовчуванням є поточний користувач Linux. Даний параметр необхідний для того, що замінити його на потрібну адресу, наприклад, на excontact@example2.com.

Також вам буде потрібно замінити останню адресу, вказану в рядку коду:

cat ~/test_message | s-nail -s 'Test email subject line' -r contact@example2.com exuser@example2.com

Після цього на вказану електронну пошту має прийти лист. Щоб перевірити надіслані повідомлення через *s-nail*, потрібно прописати *file* + *sent*, де *file* – ваша унікальна назва.

Установка і настройка Dovecot

Dovecot – це вільний IMAP- і POP3-сервер, розроблений з упором на безпеку. Нам він буде потрібний для того, щоб підключити авторизацію по протоколу SMTP.

Встановлюємо Dovecot з компонентом для роботи з СУБД:

sudo apt-get install dovecot-imapd dovecot-pop3d dovecot-mysql

Налаштовуємо спосіб зберігання повідомлень, для цього відкриємо файл:

sudo nano /etc/dovecot/conf.d/10-mail.conf

Пропишемо в ньому наступне:

mail_location = maildir:/home/mail/%d/%u/

В даному випадку повідомлення будуть зберігатися в уже відомому нам форматі Maildir.

Далі налаштовуємо слухача для аутентифікації:

sudo nano /etc/dovecot/conf.d/10-master.conf

У файлі прописуємо:

```
service auth {
  unix_listener / var / spool / postfix / private / auth {
    mode = 0666
    user = postfix
    group = postfix
  }
  unix_listener auth-userdb {
    mode = 0600
    user = vmail
    group = vmail
  }
}
```

На цьому прикладі ми налаштовуємо сервіс для аутентифікації і створюємо два прослуховувачі: */var/spool/postfix/private/auth* – для можливості Postfix використовувати авторизацію через Dovecot, *auth-userdb* – сокет для авторизації через *dovecot-lda*.

В цей же файл додаємо:

```
service stats {
   unix_listener stats-reader {
```

```
    user = vmail
    group = vmail
    mode = 0660
  }
  unix_listener stats-writer {
    user = vmail
    group = vmail
    mode = 0660
  }
}
```

Переходимо до налаштування аутентифікації в Dovecot, відкриваємо файл *10-auth.conf*:

sudo nano /etc/dovecot/conf.d/10-auth.conf

Задаємо в ньому наступні значення:

#! Include auth-system.conf.ext

! Include auth-sql.conf.ext

Відкриваємо файл *10-ssl.conf* командою *vi /etc/dovecot/conf.d/10-ssl.conf* і налаштовуємо в ньому використання шифрування:

ssl = required

ssl_cert = </etc/ssl/mail/public.pem

ssl_key = </etc/ssl/mail/private.key

Розглянемо параметри:

ssl = required – накаже Dovecot вимагати від клієнтів використання шифрування;

ssl_cert – шлях до відкритого сертифіката;

ssl_key – шлях до закритого ключа.

Додаємо автоматичне створення каталогів у файлі

sudo nano /etc/dovecot/conf.d/15-lda.conf:

lda_mailbox_autocreate = yes

Налаштовуємо підключення до бази даних. Для початку відкриваємо потрібний файл:

sudo nano /etc/dovecot/conf.d/auth-sql.conf.ext

Вставляємо в нього:

```
passdb {
  ...
```

args = /etc/dovecot/dovecot-sql.conf.ext

}

userdb {

...

args = /etc/dovecot/dovecot-sql.conf.ext

}

У цьому фрагменті ми вказали на файл, в якому будуть знаходитися настройки для отримання користувачів і паролів з БД.

Переходимо до коригування файлу з настройками роботи mySQL:

sudo nano /etc/dovecot/dovecot-sql.conf.ext

В кінці файлу додаємо:

driver = mysql

connect = host = localhost dbname = postfix user = postfix password = postfix123

default_pass_scheme = MD5-CRYPT

password_query = SELECT password FROM mailbox WHERE username = '% u'

user_query = SELECT maildir, 1024 AS uid, 1024 AS gid FROM mailbox WHERE username = '% u'

user_query = SELECT CONCAT ('/ home / mail /', LCASE (`domain`), '/', LCASE (` maildir`)), 1024 AS uid, 1024 AS gid FROM mailbox WHERE username = '% u'

Таким чином, ми змогли налаштувати запит на отримання даних з БД. Залишилося настроїти інтерфейс, на якому ми будемо слухати Dovecot:

sudo nano /etc/dovecot/dovecot.conf

У цей файл прописуємо:

*listen = **

За замовчуванням Dovecot слухає і на ipv6 (listen = *,: :). Якщо на сервері не використовується 6-а версія протоколу TCP / IP, то в логах можуть бути помилки.

Дозволяємо запуск Dovecot:

systemctl enable dovecot

Перезавантажуємося:

systemctl restart dovecot

Зміна конфігурації Postfix для Dovecot

Так як для відправки листів ми використовуємо протокол SMTP через Dovecot, нам буде потрібно внести деякі зміни в основному файлі. Відкриємо його:

sudo nano /etc/postfix/main.cf

Змінюємо в ньому наступне:

smtp_use_tls = yes
smtp_tls_security_level = may
smtpd_tls_security_level = may
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth

Також коментуємо рядки:

#virtual_mailbox_base = /home/vmail
#virtual_mailbox_maps = hash:/etc/postfix/vmaps
#virtual_minimum_uid = 1000
#virtual_uid_maps = static: 5000
#virtual_gid_maps = static: 5000

Замість них прописуємо:

virtual_transport = lmtp: unix: private/dovecot-lmtp

Відкриємо файл *sudo nano /etc/postfix/master.cf* і розкоментуємо в ньому рядок:

submission inet n - y - - smtpd

Залишилося перезавантажитися в Postfix:

sudo systemctl restart postfix

І в Dovecot:

sudo systemctl restart dovecot

Тепер можна протестувати відправку пошти. Зверніть увагу, що у Dovecot також є лог – щоб його включити, необхідно відкрити файл *sudo nano /etc/dovecot/conf.d/10-logging.conf* і внести в нього коригування:

log_path = /var/log/dovecot.log
auth_verbose = yes

Перший рядок вказує на шлях до файлу, а друга показує невдалі спроби авторизації.

Завдання до виконання

1. Встановіть та налаштуйте Postfix сервер під операційною системою Linux.
2. Протестуйте відправку та прийом поштових повідомлень.

Контрольні питання

1. Опишіть базову архітектуру сервера Postfix.
2. Опишіть процес отримання пошти сервером Postfix.
3. Опишіть процес доставки пошти сервером Postfix.
4. Перерахуйте і коротко опишіть основні допоміжні сервіси сервера Postfix.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. – СПб.: Питер, 2015. – 1120 с.

2. Шеховцов В. А. Операційні системи. – К.: Видавнича група BHV, 2005. – 576 с.

3. Колисниченко Д.Н., Аллен Питер В. LINUX: полное руководство. – СПб: Наука и Техника, 2006. – 784 с.

4. Зайцев В. Г., Дробязко І. П. Операційні системи : навч. посіб. для студ. спеціальності 123 «Комп'ютерна інженерія». – Київ: КПІ ім. Ігоря Сікорського, 2019. – 240 с. URL: https://ela.kpi.ua/bitstream/123456789/29600/1/Operatsiini_systemy.pdf

5. Погребняк Б. І., Булаєнко М. В. Операційні системи : навч. посібник. – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 104 с. URL: http://eprints.kname.edu.ua/51761/1/2017%20%D0%BF%D0%B5%D1%87.%2050%D0%9D%20%D0%9E%D0%A1_%D0%A3%D0%9F_%D0%9A%D0%9D_ua.doc.pdf

6. Смолин Илья. Установка и настройка Postfix и Dovecot на Ubuntu 20.04. URL: https://timeweb.com/ru/community/articles/ustanovka-i-nastrojka-postfix-i-dovecot-na-ubuntu

7. Изменение корневой системы. Команда chroot в Linux. URL: https://itproffi.ru/izmenenie-kornevoj-sistemy-komanda-chroot -v-linux/

8. Установка и настройка прокси-сервера Squid. URL: https://itproffi.ru/ustanovka-i-nastrojka-proksi-servera-squid/

9. Команда cut в Linux. URL: https://itproffi.ru/komanda-cut-v-linux/

10. Визуальный менеджер файлов Midnight Commander. URL: https://itproffi.ru/vizualnyj-menedzher-fajlov-midnight-commander/

11. Настройка VLAN интерфейсов в Linux. URL: https://itproffi.ru/nastrojka-vlan-interfejsov-v-linux/

12. Графические среды GNOME и KDE. URL: https://itproffi.ru/graficheskie-sredy-gnome-i-kde/

13. Настройка правил iptables в Linux. URL: https://itproffi.ru/nastrojka-pravil-iptables-v-linux/

14. Настройка сети в Linux – команда ifconfig. URL: https://itproffi.ru/nastrojka-seti-v-linux-komanda-ifconfig/

15. Редактор VIM: как пользоваться, инструкция команд. URL: https://itproffi.ru/redaktor-vim-kak-polzovatsya-instruktsiya-komand/

16. Установка и настройка FTP-сервера в Linux. URL: https://itproffi.ru/ustanovka-i-nastrojka-ftp-servera-v-linux/

17. Установка и настройка сервера SSH в Linux. URL: https://itproffi.ru/ustanovka-i-nastrojka-servera-ssh-v-linux/

18. 50 основных команд Linux для новичков. URL: https://timeweb.com/ru/community/articles/komandy-linux

19. Бухтеев Виктор. Права суперпользователя root в Linux: полный обзор. URL: https://timeweb.com/ru/community/articles/root-v-linux

20. Командная строка Ubuntu: основные команды bash. URL: https://timeweb.com/ru/community/articles/komandnaya-stroka-ubuntu-osnovnye-komandy-bash-1

21. Харченко В. П., Знаковська Є. А., Бородін В. А. Операційні системи та системи програмування : навч. посіб. – К.: Вид-во Нац. авіац. ун-ту «НАУ-друк», 2012. – 360 с.

22. Авраменко В. С., Авраменко А. С. Основи операційних систем : навчальний посібник. – Черкаси: ЧНУ імені Богдана Хмельницького, 2018. – 524 с.

23. Семеренко В. П., Крилик Л. В. Операційна система Linux : навчальний посібник. – Вінниця: ВНТУ, 2006. – 88 с.

Навчальне видання

ГУЧЕК Петро Йосипович
ЛИТВИНЕНКО Олена Іванівна

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з курсу «ОПЕРАЦІЙНІ СИСТЕМИ»

Комп'ютерне верстання *В. В. Москаленко*
Коректор *О. Є. Вакула*

Формат 60×84/16. Ум. друк. арк. 6,6. Тираж 100 прим. Вид. № 31. Зам. № 1801-08.
Видавець і виготівник Національний університет кораблебудування
імені адмірала Макарова
просп. Героїв України, 9, м. Миколаїв, 54025
E-mail : publishing@nuos.edu.ua
Свідоцтво суб'єкта видавничої справи ДК № 6402 від 19.09.2018 р.