МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

АДМІНІСТРУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ

для студентів першого (бакалаврського) рівня вищої освіти
спеціальності 123 "Комп'ютерна інженерія"

Затверджено
на засіданні кафедри
електронних обчислювальних машин.
Протокол № 1 від 28 серпня 2020 р.

Львів 2020

Адміністрування комп'ютерних систем і мереж: методичні вказівки до лабораторних робіт для студентів першого (бакалаврського) рівня вищої освіти спеціальності 123 "Комп'ютерна інженерія" / Укл. М. О. Хомуляк – Львів: Видавництво Національного університету "Львівська політехніка", 2020. – 64 с.

Укладач Хомуляк М. О., ст. викладач

Відповідальний за випуск Березко Л. О., канд. техн. наук, доцент

Рецензенти Ваврук Є. Я., канд. техн. наук, доцент,
Юрчак І. Ю., канд. техн. наук, доцент

ЗМІСТ

ВСТУП

Методичні вказівки до лабораторних робіт укладені відповідно до навчальної програми з дисципліни “Адміністрування комп’ютерних систем і мереж” та включають шість лабораторних робіт.

Виконання лабораторних робіт побудовано на дослідженні команд та утиліт, характерних для організації роботи комп’ютерної мережі. При цьому опрацьовуються такі питання:

- ознайомлення з особливостями архітектури та основними компонентами операційної системи Linux;
- ознайомлення з програмним середовищем та командами загального призначення;
- вивчення структури файлової системи та операцій з файлами;
- вивчення команд керування процесами та їх взаємодією;
- вивчення прав доступу і способи захисту файлів;
- робота з користувачами;
- ознайомлення з організацією та адмініструванням мережі.

Лабораторні роботи передбачають використання комп’ютерної бази, оснащеної операційною системою Linux. Методичні матеріали призначені для допомоги студентам у підготовці до лабораторних занять, дослідженні особливостей застосування команд та утиліт, виконанні індивідуальних завдань, захисті отриманих результатів.

ЛАБОРАТОРНА РОБОТА № 1

Особливості архітектури та основні компоненти операційної системи Linux

МЕТА РОБОТИ: Ознайомитись з архітектурою операційної системи Linux та її основними компонентами.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Операційна система Linux - це набір програм, що керує комп'ютером, здійснює зв'язок між користувачем і комп'ютером та надає користувачу інструментальні засоби, щоб допомогти йому виконати роботу. Їй притаманна легкість, ефективність і гнучкість програмного забезпечення.

Майже три десятиліття існування Linux – дуже великий строк для операційної системи. Сміливо можна сказати, що вона повністю витримала перевірку часом. На кожному етапі свого розвитку операційна система Linux вирішувала певні завдання, і сьогодні, незважаючи на появу простіших і зручніших, з погляду адміністрування, систем, Linux міцно займає місце серед лідерів. Причини популярності Linux:

1. Код системи написаний мовою високого рівня С, що зробило її простою для розуміння, змін і перенесення на інші платформи. Написана мовою С система мала на 20…40% більший розмір, а продуктивність її була на 20% нижчою від аналогічної системи, написаної на асемблері. Однак ясність і переносимість, а в результаті - і відкритість системи зіграли вирішальну роль у її популярності. Можна сміливо сказати, що Linux є однією з найбільш відкритих систем. Незважаючи на те, що частина версій Linux поставляється не у початкових текстах, а у вигляді бінарних файлів, система залишається такою, що можна легко расширювати і налаштовувати.

2. Linux – багатозадачна та розрахована на багато користувачів система зі широким спектром послуг. Один потужний сервер може обслуговувати запити великої кількості користувачів. При цьому є необхідність в адмініструванні лише однієї системи. Система може виконувати різні функції – працювати як обчислювальний сервер, що обслуговує сотні користувачів, як сервер бази даних, як мережний сервер, що підтримує найважливіші сервіси мережі (telnet, ftp, электронну пошту, службу імен DNS тощо), або навіть як мережний маршрутизатор.

3. Наявність стандартів. Незважаючи на різноманіття версій Linux, основою всього сімейства є принципово однакова архітектура і ряд стандартних інтерфейсів: Досвідчений адміністратор без зусиль зможе обслужити іншу версію системи, для користувачів перехід на іншу версію і зовсім може виявитися непомітним.

4. Простий, але потужний модульний інтерфейс користувача. За наявності набору утиліт, кожна з яких вирішує вузьку спеціалізовану задачу, можна конструювати з них складні комплекси.

5. Використання єдиної ієрархічної файлової системи, що легко обслуговується. Файлова система – це не тільки доступ до даних, що зберігаються на диску. Через уніфікований інтерфейс файлової системи здійснюється доступ до терміналів, принтерів, магнітних стрічок, мережі та навіть до пам'яті.

6. Дуже велика кількість прикладних програм, у тому числі таких, що вільно розповсюджуються, починаючи від найпростіших текстових редакторів і закінчуючи потужними системами керування базами даних.

Архітектура Linux

Дворівнева модель операційної системи Linux показана на рис. 1.

Рис. 1. Модель операційної системи Linux

У центрі перебуває ядро системи (kernel). Ядро безпосередньо взаємодіє з апаратною частиною комп'ютера, ізолюючи прикладні програми від особливостей її архітектури. Ядро має набір послуг, що надаються прикладним програмам. До послуг ядра відносяться операції введення/виведення (відкриття, читання, запис і керування файлами), створення і керування процесами, їх синхронізації та міжпроцесорної взаємодії. Всі прикладні програми запитують послуги ядра за допомогою системних викликів.

Другий рівень становлять програми або задачі, як системні, що визначають функціональність системи, так і прикладні, що забезпечують інтерфейс користувача Linux. Однак, незважаючи на зовнішню різнорідність прикладних програм, схеми їхньої взаємодії з ядром однакові.

Ядро системи

Ядро забезпечує базову функціональність операційної системи: створює процеси й керує ними, розподіляє пам'ять і забезпечує доступ до файлів і периферийних пристроїв.

Взаємодія прикладних задач з ядром відбувається за допомогою стандартного інтерфейсу системних викликів. Інтерфейс системних викликів є набором послуг ядра й визначає формат запитів на послуги. Процес запитує послугу за допомогою системного виклику певної процедури ядра, зовні схожого на звичайний виклик бібліотечної функції. Ядро від імені процесу виконує запит і повертає процесу необхідні дані.

У наведеному нижче прикладі програма відкриває файл, зчитує з нього дані й закриває цей файл. При цьому операції відкривання (open), читання (read) і закривання (close) файлу виконуються ядром на запит задачі, а функції *open(2}, read(2}* і *close(2}* є системними викликами.

```
main()
(
     int fd;
 char buf[80];
/* Відкриваємо файл - отримуємо посилання (файловий
дескриптор) fd*/
fd = open ("filel", O_RDONLY);
/* Зчитуємо в буфер buf 80 символів */
read(fd, buf, sizeof(buf));
/* Закриваємо файл */
close(fd);
```

Структура ядра показана на рис 2.

Ядро складається із трьох основних підсистем:

1. Файлова підсистема.
2. Підсистема керування процесами і пам'яттю.
3. Підсистема вводу/виводу.

Файлова підсистема

Файлова підсистема забезпечує уніфікований інтерфейс доступу до даних, розташованих на дискових накопичувачах, і до периферійних пристроїв. Ті самі функції *open(2)} read(2), write(2)* можуть використовуватися як при читанні або запису даних на диск, так і при виведенні тексту на принтер або

термінал.

Файловая підсистема контролює права доступу до файлу, виконує операції розміщення й видалення файлу, а також виконує запис/читання даних файлу. Оскільки більшість прикладних функцій виконується через інтерфейс файлової системи (в тому числі і доступ до периферійних пристроїв), права доступу до файлів визначають привілеї користувача у системі.

Файлова підсистема забезпечує перенаправлення запитів, адресованих периферійним пристроям, що відповідають модулям підсистеми вводу/виводу.

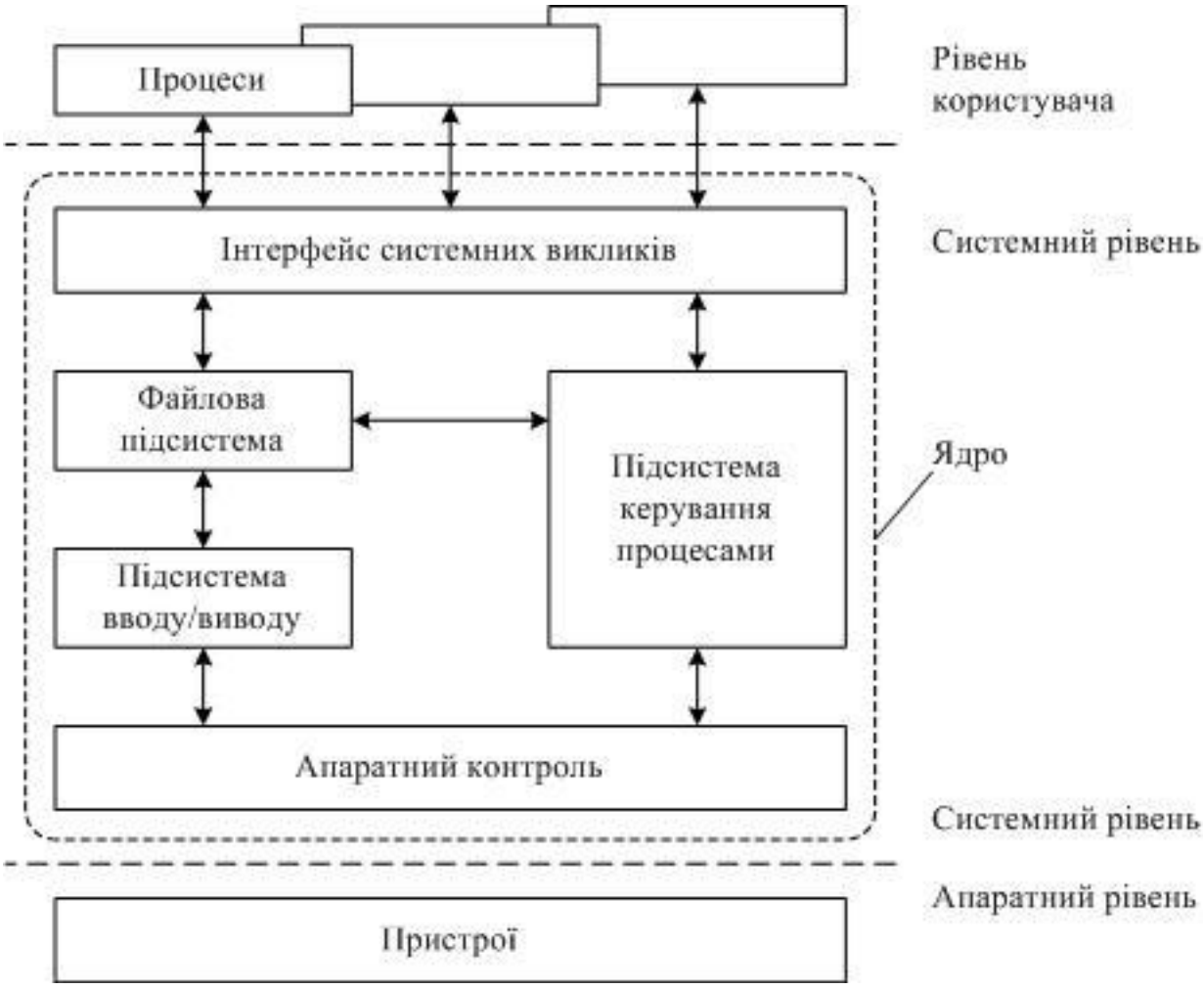

Рис. 2. Внутрішня структура ядра Linux

Підсистема керування процесами

Запущена на виконання програма породжує в системі один або більше процесів (або задач). Підсистема керування процесами контролює:

– Створення й видалення процесів.

– Розподіл системних ресурсів (пам'яті, обчислювальних ресурсів) між процесами.

– Синхронізацію процесів.

– Міжпроцесну взаємодію.

Очевидно, що в загальному випадку число активних процесів перевищує

число процесорів комп'ютера, але в кожен конкретний момент часу на кожному процесорі може виконуватися тільки один процес. Операційна система керує доступом процесів до обчислювальних ресурсів, створюючи відчуття одночасного виконання декількох задач.

Спеціальна задача ядра, що називається розпорядником або планувальником процесів (scheduler), вирішує конфлікти між процесами в конкуренції за системні ресурси (процесор, пам'ять, пристрої вводу/виводу). Планувальник запускає процес на виконання, стежачи за тим, щоб процес монопольно не захопив поділювані системні ресурси. Процес звільняє процесор, очікуючи тривалої операції введення/виведення, або після закінченні кванта часу. У цьому випадку планувальник вибирає наступний процес із найвищим пріоритетом і запускає його на виконання.

Модуль керування пам'яттю забезпечує розміщення оперативної пам'яті для прикладних задач. Оперативна пам'ять є дорогим ресурсом, і, як правило, її рідко буває “занадто багато”. У випадку, якщо для всіх процесів недостатньо пам'яті, ядро переміщує частини процесу чи декількох процесів у вторинну пам'ять (як правило, у спеціальну область твердого диска), звільняючи ресурси для процесу, що виконується. Всі сучасні системи реалізують так звану віртуальну пам'ять: процес виконується у власному логічному адресному просторі, що може значно перевищувати доступну фізичну пам'ять. Керування віртуальною пам'яттю процесу також входить до завдань модуля керування пам'яттю.

Модуль міжпроцесної взаємодії відповідає за повідомлення процесів про події за допомогою сигналів і забезпечує можливість передачі даних між різними процесами.

Підсистема вводу/виводу

Підсистема вводу/виводу виконує запити файлової підсистеми і підсистеми керування процесами для доступу до периферійних пристроїв (дисків, магнітних стрічок, терміналів тощо). Вона забезпечує необхідну буферизацію даних і взаємодіє з драйверами пристроїв – спеціальними модулями ядра, що безпосередньо обслуговують зовнішні пристрої.

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ КОМАНД

1. Команда man

man *параметри команда*

Команда **man** форматує і виводить “Сторінки посібника”. “Сторінки посібника” є офіційною документацією з операційної системи Linux і мають жорстко заданий формат. “Сторінки посібника” дуже корисні для пошуку інформації про незрозумілі параметри чи команди.

Зазвичай при запуску **man** треба вказати команду, про яку необхідно отримати інформацію.

Параметри:

-a Виведення усіх “Сторінок посібника” із заданою назвою, а не лише першої.

-c Переформатування “Сторінок посібника”, навіть якщо вже існує відформатований варіант.

-C *(файл)* Використання зазначеного файлу як файлу конструкції; за замовчуванням використовується файл **/usr/lib/man.config**.

-d Виведення налагоджувальної інформації без “Сторінок посібника”.

-D Виведення налагоджувальної інформації і “Сторінок посібника”.

-f Виведення короткого опису команди.

-h Виведення короткої довідки з команди **man**.

-k Емуляція команди **apropos**.

-m *(система)* Пошук “Сторінок посібника” у заданій системі.

-M *(шлях)* Встановлення шляху для пошуку “Сторінок посібника”. За замовчуванням використовується значення змінної оточення MANPATH. Якщо вона не встановлена, використовується шлях, заданий у файлі конфігурації **/usr/lib/man.config**.

-p *(рядок)* Використання вказаних препроцесорів перед запуском **nroff** чи **groff.** Цей параметр практично даремний при роботі з Linux.

-P *(команда)* Використати вказану команду для посторінкового виведення. За замовчуванням використовується **/usr/bin/ less -is**.

-S *(список_розділів)* Пошук “Сторінок посібника” у зазначених розділах.

-t Використання для форматування “Сторінок посібника” команди **/usr/bin/groff -Tps -mandoc**.

-w Виведення розташування “Сторінок посібника”, але не самих “Сторінок”.

-W Виведення розташування “Сторінок посібника”, але не самих “Сторінок”. Шлях до кожної наступної “Сторінки” виводиться в окремому рядку.

Споріднені команди:

apropos Пошук заданого ключового слова у базі даних **whatis**.

manpath Встановлення або виведення шляху, що використовується командою **man** для пошуку “Сторінок посібника”.

whatis Виведення рядка інформації на задану тему.

xman Виведення “Сторінок посібника” в X-вікні.

2. Команда date

date *параметри +формат*

date *параметри рядок (привілейовані користувачі)*

Команда **date** дозволяє вивести поточну дату і час в одному з безлічі форматів. Привілейовані користувачі (наприклад, *root)* також можуть використовувати цю команду для встановлення дати і часу.

Параметри:

+*формат* Виведення дати в одному з перерахованих нижче форматів.

-s Встановлення дати і часу. Цей параметр доступний тільки привілейованим користувачам.

-u Виведення дати і часу за Грінвічем.

Формати:

%a Скорочена назва дня тижня (Sun, Mon і т.д.).

%A Назва дня тижня (Sunday, Monday, і т.д.).

%b Скорочена назва місяця (Jan, Feb, і т.д.). Те ж, що і %h.

%B Назва місяця (January, February і т.д.).

%c Дата і час у місцевому часовому поясі.

%d Число місяця у вигляді двозначного числа (01-31).

%D Дата у форматі *mm/dd/yy (mm* — номер місяця, *dd*— число, *yy*— дві останні цифри року).

%e Число місяця (1-31).

%h Скорочена назва місяця (Jan, Feb і т.д.). Те ж, що і **%b.**

%H Година у 24-годинному форматі (00-23).

%I Година у 12-годинному форматі (00-12).

%j Юліанська дата (номер дня у році, 1-365).

%k Година у 24-годинному форматі без ведучих нулів (0-23).

%l Година у 12-годинному форматі без ведучих нулів (0-12).

%m Номер місяця як двозначне число (01-12).

%M Хвилини (0-59).

%n Символ нового рядка.

%p Використання символів a.m. і p.m. замість використовуваних за замовчуванням AM і PM.

%r Час у форматі *hh:mm:ss* AM/PM **(***hh* — години, *mm* — хвилини, *ss* — секунди).

%s Кількість секунд, що пройшла від “початку епохи”, 1970-01-01 00:00:00 UTC.

%S Секунди (0-59).

%t Символ табуляції.

%T Час у форматі *hh:mm:ss (hh* — години, *mm* — хвилини, *ss* — секунди).

%U Число місяця (01-31).

%w Номер дня тижня (неділя — 0).

%W Номер тижня (0-51); першим днем тижня вважається понеділок.

%X Час у національному форматі.

%x Дата у національному форматі.

%y Дві останні цифри року (наприклад, 98).

%Y Рік (наприклад, 2010).

%Z Часовий пояс.

Приклади:

```
$date
```
(На екран виводиться поточна дата і час).

```
$date –u
```
(На екран виводиться поточна дата і час за Грінвічем).

```
$date +%A
```
(На екран виводиться назва дня тижня).

Встановлення дати

```
$date 1115094806
```

Привілейовані користувачі можуть встановлювати дату і час. Дата може бути зазначена в числовому чи в нечисловому форматі. При заданні дати в числовому форматі використовується рядок *MMddhhmmyy (MM*— місяць *dd* — число, *hh* — година, *mm* — хвилини, *yy* — дві останні цифри року).

Ця команда встановлює системний час 09:48, дату – 15 листопада 2006 року. Лише привілейовані користувачі можуть змінювати системну дату і час.

3. Команда who

who *параметри файл*

Виведення списку користувачів, підключених у даний момент до системи.

Параметри:

am I Виведення інформації про користувача, що запустив цю команду.

-a Використовувати всі зазначені нижче параметри.

-b Виведення дати і часу останнього перезавантаження системи.

-d Виведення списку користувачів, відключених через тривалу неактивність.

-H Виведення на початку списку заголовків стовпчиків.

-I Виведення списку ліній, доступних для входу в систему.

-n*n* Виведення в одному рядку інформації про *n* користувачів.

-p Виведення списку процесів, запущених процесом **init** і все ще активних.

-q Короткий формат; виводяться тільки системні ідентифікатори користувачів.

-r Виведення рівня запуску системи.

-s Виведення системного ідентифікатора користувача, термінала і часу неактивності (формат, що використовується за замовчуванням).

-t Виведення часу, коли останній раз за допомогою команди **clock** налаштовувався системний годинник.

-T Виведення стану кожного термінала:

+ Термінал доступний для виведення всім користувачам.

- Термінал доступний для виведення лише системному адміністратору.

? Помилка при визначенні стану термінала.

-u Виведення часу неактивності для кожного термінала.

Споріднені команди:

date Виведення дати і часу.
login Вхід у систему.
mesg Встановлення прав доступу до термінала.
rwho Виведення списку користувачів на машинах локальної мережі.

4. Команда cal

cal *параметри №місяця рік*

Виведення календаря. Якщо параметри не вказані, виводиться календар на поточний місяць.

Параметри:

-m Виведення понеділка як першого дня тижня.
-j Виведення дат юліанського календаря.
-y Виведення календаря на поточний рік.

5. Команда sleep

sleep *секунди*

Вказує системі почекати задане число секунд, перш ніж виконувати наступну команду. Ця команда зручна при написанні сценаріїв. Число задає час в секундах; якщо за числом іде суфікс **m**, **h** або **d,** то число задає час у хвилинах, годинах або добах відповідно.

Комбінації клавіш

<Ctrl+c> – знищення поточного процесу.
<Ctrl+j> – вставляння у командний рядок вибраного файлу.
<Ctrl+e>/<Ctrl+x> – пересування по хронології команд вгору/вниз.
<Ctrl+p> – перехід у режим командного рядка (повернення назад).
<Ctrl+t> – виділення вибраного файлу.
<Ctrl+s>/<Ctrl+q > – зупинка/відновлення виведення на екран.
<Ctrl+b> – показ хронології команд.
<Ctrl+k> – пошук файлу.
<Ctrl+l> – статус (стан).
<Ctrl+r> – перечитування.
<Ctrl+]> – перемальовування екрану.
<Ctrl+i> – перехід на іншу панель.
<Ctrl+u> – зміна панелей місцями.
<Ctrl+f> – повноекранний режим.
<Ctrl+w> – подвійна ширина вибраної панелі.

ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

1. Зареєструватись у системі.
2. Ознайомитися з призначенням та особливостями застосування рекомендованих команд і комбінацій клавіш.
3. Дослідити виконання команд з параметрами, які збігаються у методичних вказівках та на сторінках системного електронного посібника.
4. Скласти і захистити звіт.

ЗМІСТ ЗВІТУ

1. Номер, назва і мета лабораторної роботи.
2. Призначення і стислий опис команд з окремими параметрами.
3. Результати досліджень команд та комбінацій клавіш (запуск і реакція системи).
4. Висновки.

ЛАБОРАТОРНА РОБОТА № 2

Програмне середовище операційної системи Linux. Команди загального призначення

МЕТА РОБОТИ: Ознайомитись з програмним середовищем та командами загального призначення ОС Linux.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Після входу в систему користувач потрапляє у певне програмно-апаратне середовище, яке забезпечує виконання поставлених перед ним задач. Всередині операційної системи Linux існує багато різних підсередовищ. Разом вони утворюють загальну картину, у вигляді якої ми уявляємо собі Linux.

Різноманіття середовищ на рис.3 демонструє різні рівні, що функціонують всередині комп'ютера. Нижній шар – це стартова точка, від якої різноманіття росте нагору. Кожний рівень будується на попередньому й використається для підтримки рівня, розташованого над ним. Для кожного вищого рівня середовище об'ємніше і “віртуальніше” у тому розумінні, що має місце менше умовних обмежень. Верхні рівні використають для своєї роботи нижні і, таким чином, приховують подробиці, необхідні для роботи цих нижніх рівнів. Можна створити моделі високого рівня, які працюють на машині нижчого рівня, не знаючи нічого про нижні рівні.

Рівень 7	**Командні файли (scripts)**
Рівень 6	**Прикладні програми, інтерпретатор команд, генератори мови**
Рівень 5	**Компілятор**
Рівень 4	**Операційна система**
Рівень 3	**Ядро**
Рівень 2	**Умовна машина, асемблер**
Рівень 1	**Мікропрограми**
Рівень 0	**Логічні схеми, апаратні засоби**

Рис.3. Різноманіття середовищ комп'ютера

Рівень 0. (Логічні схеми, апаратні засоби). На найнижчому рівні перебувають апаратні засоби й логічні кола. Цей рівень визначає спосіб

зберігання та обробки даних у всіх апаратних засобах. На цьому рівні компонентами є центральний процесор, пам'ять, мікросхеми підтримки і системна шина. Хоча прогрес на цьому рівні триває, це викликає дуже малі зміни на верхньому шарі піраміди. Філософія системи Linux полягає у тому, щоб ізолювати низькорівневий апаратний шар і забезпечити до нього однакові інтерфейси, які не мають потреби у змінах “нагорі”. Верхній шар навіть не повинен знати про нижній шар.

Рівень 1. (Мікропрограми). Цей рівень багато в чому схожий на мову програмування. Він є інструментом, який використовує архітектор системи для створення “рідної” машинної мови. Машинна мова повідомляє апаратурі, яку конкретну команду слід виконати.

Рівень 2. (Умовна машина, асемблер). Даний рівень забезпечує трансляцію з мнемонік мови асемблера в коди операцій і дані машинної мови. Мова асемблера - це деяка англо-подібна нотація, що полегшує людині розуміння й керування роботою комп'ютерів. Умовна машина підтримується асемблером. Асемблер може перетворювати ідеї більш високого рівня в послідовності чисел, які можуть бути потім виконані. Поряд з асемблером, застосовуються моделі, що допомагають використати апаратуру комп'ютера. Тут можна визначити такі поняття, як стеки, вектори переривань і периферійне введення/виведення.

Рівень 3. (Ядро). Ядро є концепцією, яку можна реалізувати програмно на умовній машині. Ядро надає середовище, що підтримує ще більші абстракції, ніж ті, що розглядалися дотепер. Двома найважливішими абстракціями на рівні ядра є керування процесами для мультипрограмування і багатозадачності, та файлова система, яка керує зберіганням, форматом, пошуком файлів тощо. Коли ці дві області переплітаються, можна говорити про базову функцію розрахованої на багато користувачів машини і ядро операційної системи.

Однієї з найбільш важливих областей, якими керує ядро, є безпека. Перевірки ідентифікації користувача виконуються в системних викликах всередині ядра. Певні механізми використаються ядром для керуваннння безпекою файлів, пристроїв, пам'яті та процесів. Єдиний спосіб відключити механізми безпеки полягає у зміні вихідного коду ядра і перекомпіляції всієї системи.

Рівень 4. (Операційна система). Даний рівень будується на ядрі, щоб створити повне операційне середовище. Потребу в додаткових функціях системи можна задовольнити створенням автономних програм, що мають конкретне призначення. Таким чином, сукупність всіх специфічних функцій визначає операційну систему.

Рівень 5. (Компілятор). Компілятор - це інструмент, побудований на операційній системі для подальшої розробки досконаліших і могутніших середовищ. Нові середовища можуть припускати ще більші абстракції, ніж на нижньому рівні, і робити більше допущень про те, що вже існує. Це робить

можливими символічні конструкції більш високого рівня, такі як структури даних й керуючі структури. Результатом є прикладна програма.

Рівень 6. (Прикладні програми, інтерпретатор команд, генератори мови). Прикладні програми можуть означати найрізноманітніші програмні продукти. Можна припустити, що будь-яка програма, яка зроблена за допомогою компілятора, є прикладною програмою. Прикладами можливих прикладних програм є наступне покоління мов, інтерпретаторів і генераторів прикладних програм. Інтерпретатор - це програма, написана розповсюдженою мовою високого рівня, що може декодувати і виконувати інший синтаксис (або мову). Прикладом, що може бути цікавим у системі Linux, є командний процесор shell. Це програма мовою С, створена для читання і виконання команд, записаних за правилами синтаксису, визначених командним процесором shell.

Генератор прикладних програм - це програма, написана мовою високого рівня. Вона призначена для одержання достатньої інформації від користувача про його прикладну програму і може використати компіляторну мову, наприклад С, для написання прикладної програми, яка реалізує те, що потрібно. Користувач нічого не програмує. Виходом генератора є робоча програма.

Рівень 7. (Командні файли). Цей верхній рівень є мовою, що інтерпретує програма /bin/sh (у випадку командного процесора Bourne shell). Її синтаксис підтримує повна мова програмування. Хоча ця мова позбавлена ряду вбудованих структур і функцій сучасної мови високого рівня, вона має все необхідне для написання корисних програм. Великою перевагою є те, що мові командного процесора доступні у вигляді зовнішніх функцій будь-які засоби, утиліти та програми, які є в системі Linux. Це означає, що алгоритми, які можуть вимагати сто або більше рядків мовою низького рівня типу С, мова командного процесора може виразити у двадцять рядків, зрозуміло, за рахунок втрати продуктивності.

В операційній системі Linux об’єкти, які запускаються з командного рядка, можуть бути:

– функціями, що визначаються користувачем;

– вбудованими командами інтерпретатора;

– виконуваними файлами – прикладними програмами та утилітами.

Виконуваний файл, у свою чергу, може бути “двійковим”, тобто програмою у кодах ЕОМ, текстовим командним файлом оболонки ОС чи байт-кодом для віртуальної машини Java. Ознакою виконуваного файлу є встановлений для нього спеціальний атрибут “executable”, який дозволяє його виконання як програми.

Програми є командами, зовнішніми відносно оболонки. Деякі дії не можуть бути виконані зовнішніми командами, їх обов’язково повинна виконати сама оболонка. Команди, код яких міститься у виконуваному файлі оболонки, називаються вбудованими, або внутрішніми.

Якщо набрана користувачем назва команди є назвою вбудованої команди, вона негайно буде виконана. Якщо ж назва не збігається з назвами вбудованих

команд, оболонка здійснить пошук виконуваного файлу програми з вказаною назвою у каталогах файлової системи.

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ КОМАНД

1. Команда time

time *параметри команда*

Запуск зазначеної команди з підрахунком часу, витраченого на її виконання (загальний час, час користувача, системний час). Має параметри, що дозволяють вивести загальну кількість прочитаних з диска і записаних на диск блоків, рівень зайнятості системи та іншу інформацію.

Приклад:

```
$time ls
```

(Буде запущена команда **ls**, яка виводить список файлів у поточному каталозі, після чого **time** виведе час, що зайняло виконання команди **ls**.).

Параметри:

-o Виведення кількості використаних блоків і переданих символів.

-p *додаткові_параметри* Виведення інформації про процес, запущений зазначеною *командою,* заданої одним або декількома з наступних параметрів.

-f Виведення стану прапорців розгалуження/завершення і коду завершення.

-h Виведення відношення часу, витраченого процесором, до загального часу.

-m Виведення середнього обсягу пам'яті, зайнятого процесом (цей параметр встановлений за замовчуванням).

-r Виведення інформації про відносне завантаження процесора.

-t Виведення часу, витраченого процесором, і часу, витраченого системою.

-s Виведення інформації про загальне завантаження системи під час роботи команди.

2. Команда cat

cat *параметри файли*

Команда **cat** — одна із найкорисніших команд операційної системи, оскільки вона дозволяє виконувати ряд базових операцій з файлами. У найпростішому варіанті команда **cat** читає зазначений файл і виводить його вміст на стандартний вивід (на екран, якщо стандартний вивід не перенаправлений). Команда **cat** може використовуватися у поєднанні зі символом перенаправлення > для об'єднання декількох файлів в один, а також зі символом перенаправлення >> для дописування файлів у кінець існуючого. Нарешті, команда **cat** може використовуватися для створення нового текстового файлу.

Параметри:

-A, --show-all Виведення усіх недрукованих і керуючих символів, за винятком символів переведення каретки і символів табуляції. Крім того, наприкінці кожного рядка виводиться символ $, і замість символу табуляції виводиться ^I. Цей параметр аналогічний параметру **-vET**.

-e,-E,--show-ends Виведення у кінці кожного рядка знака $.

-n Нумерація рядків.

-s Заборона виведення незаповнених рядків.

-t Виведення символів табуляції як ^I та символів перегляду сторінки як ^L.

-T, --show-tabs Виведення символів табуляції як ^I.

-u Заборона виведення будь-якої інформації на екран. Цей параметр існує для сумісності зі старими Linux -сценаріями.

-v Виведення всіх недрукованих та керуючих символів, за винятком символів переведення каретки та символів табуляції.

Приклад:

```
$cat file1 file2 >file3
```

(Ця команда об'єднує вміст файлів **file1** і **file2** і записує у **file3**).

Функціональний еквівалент у командному рядку операційної системи Microsoft Windows: **type**.

3. Команда more

more *параметри файли*

Виведення файлу на екран частинами. Розмір виведеної частини дорівнює висоті екрана. Для того щоб побачити наступну частину, треба натиснути пробіл. Щоб перервати виконання команди слід натиснути клавішу q.

Приклад:

```
$more bigfile
```

(Ця команда виводить на екран файл **bigfile,** розбиваючи його на частини, що цілком поміщаються на екрані).

Параметри:

-c Очищення екрана перед виведенням наступної порції тексту. Це швидше, ніж прокручування екрана.

-d Виведення у нижній частині екрана рядка з короткими інструкціями.

-f Розбивання довгих рядків на декілька, що поміщаються на екрані. Відповідно, зменшується кількість початкових рядків, що виводяться на екран за один раз.

-1 Ігнорувати символ протягання сторінки (^L) в останньому рядку сторінки.
-r Дозвіл виведення керуючих символів.
-s Ігнорувати порожні рядки, що повторюються.
-u Ігнорувати атрибути тексту, такі як підкреслення.
-w Очікування натиснення клавіші перед завершенням роботи.
-n Встановлення розміру виведеної за один раз сторінки тексту в *n* рядків.
+n Виведення тексту, починаючи з рядка номер *n*.

Керуючі клавіші:

При перегляді тексту можна використовувати такі клавіші:
f Перехід до наступної сторінки.
n Перехід до наступного файлу.
p Перехід до попереднього файлу.
q Завершення роботи.

Споріднені команди:

pg Розбиття файлу на сторінки.
less Перегляд вмісту файлу у будь-якому напрямку.

4. Команда ed

ed *параметри файли*

Простий текстовий редактор. Може працювати у двох режимах – командному чи введення тексту. У перший режим користувач потрапляє за замовчуванням. У ньому будь-яка клавіша сприймається як команда.

Найуживаніші команди редактора ed:

a Перехід до режиму введення (додавання) рядків символів.
. Завершення режиму введення (єдина у рядку крапка).
w *файл* Запис введених рядків до файлу.
q Вихід з редактора.
Q Вихід з редактора без запису.

Споріднені команди:

crypt Шифрування файлів.
vi Текстовий редактор.

5. Команда wc

wc *параметри файли*

Підрахунок кількості рядків, слів і символів в одному чи декількох текстових файлах. Команда послідовно виводить кількість рядків, слів та символів.

Параметри:

-bytes Підрахунок кількості байтів.
-c Підрахунок кількості символів.

--chars Підрахунок кількості символів.
-l Підрахунок кількості рядків.
--lines Підрахунок кількості рядків.
-w Підрахунок кількості слів, виконується за замовчуванням.
--words Підрахунок кількості слів, виконується за замовчуванням.

Приклад:

```
$wc textfile
324
```

6. Команда echo

echo *параметри рядок*

Команда **echo** виводить текст або значення змінних на стандартний вивід (зазвичай на екран, якщо не використовується перенаправлення виведення). Насправді існують три варіанти команди **echo:** команда Linux **/bin/echo,** а також команди **echo** інтерпретаторів командного рядка C shell та Bourne Again shell. Ці три варіанти практично однакові; єдина істотна відмінність полягає у тому, що команда C shell не підтримує параметр **-n** і виведення керуючих символів.

Параметр:

-n Не виводити в кінці символ нового рядка.
-e Увімкнути інтерпретацію керуючих символів.

Керуючі символи:

\a Звуковий сигнал.
\b Повернення на крок.
\c Не виводити символ нового рядка.
\f Протягання сторінки.
\n Переведення рядка.
\r Повернення каретки.
\t Горизонтальна табуляція.
\v Вертикальна табуляція.
\\ Зворотна похила риска (\).
nnn Символ з вісімковим кодом *nnn*.

Приклади:

```
$echo "Good afternoon!"
```

(Ця команда виводить на екран рядок **Good afternoon!**).

```
$echo "We are testing the printer" | lp
```

(Ця команда виводить на принтер рядок **We are testing the printer**).

7. Команда tty

tty *параметри*

Виведення інформації про термінал, з'єднаний зі стандартним вводом. Ця команда часто використовується в сценаріях для перевірки того, що сценарій викликається з термінала.

Параметри:

-a Виведення усієї доступної інформації. Цей параметр є не у всіх системах.

-s Виведення лише коду повернення: 0 (термінал), 1 (не термінал) або 2 (помилковий параметр).

Споріднена команда:

stty Налаштування режиму роботи термінала.

ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

1. Зареєструватись у системі.
2. Ознайомитися з призначенням та особливостями застосування рекомендованих команд.
3. Дослідити виконання команд з параметрами, які збігаються у методичних вказівках та на сторінках системного електронного посібника.
4. Скласти і захистити звіт.

ЗМІСТ ЗВІТУ

1. Номер, назва і мета лабораторної роботи.
2. Призначення і стислий опис команд з окремими параметрами.
3. Результати досліджень команд (запуск і реакція системи).
4. Висновки.

ЛАБОРАТОРНА РОБОТА № 3

Робота з файловою системою ОС Linux

МЕТА РОБОТИ: Вивчити структуру файлової системи ОС Linux. Навчитись роботі з файлами.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Файлова система Linux характеризується:

- ієрархічною структурою,
- погодженою обробкою масивів даних,
- можливістю створення і видалення файлів,
- динамічним розширенням файлів,
- захистом інформації у файлах,
- трактуванням периферійних пристроїв (таких як термінали і стрічкові пристрої) як файлів.

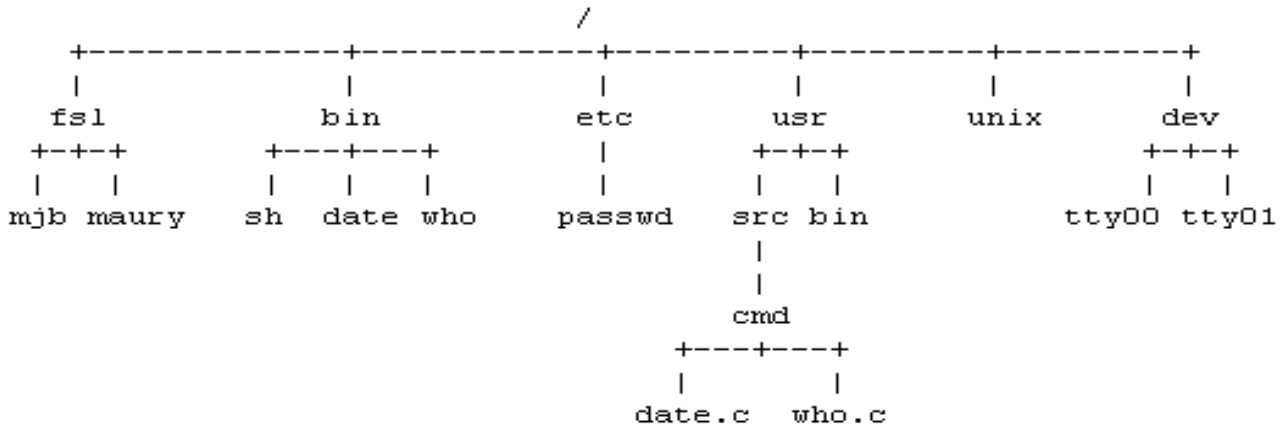

Рис. 4. Приклад деревоподібної структури файлової системи

Файлова система організована у вигляді дерева з однією вихідною вершиною, що називається коренем (записується: "/"); кожна вершина у деревоподібній структурі файлової системи, крім листків, є каталогом файлів, а файли, що відповідають дочірнім вершинам, є або каталогами, або звичайними файлами, або файлами пристроїв. Назві файлу передує вказування шляху пошуку, який описує місце розташування файлу в ієрархічній структурі файлової системи. Назва шляху пошуку складається з компонентів, розділених між собою похилою рискою (/); кожен компонент є набором символів, що складають назву вершини (файлу), що є унікальною для каталогу (попереднього компонента), у якому вона утримується. Повна назва шляху пошуку починається з вказування похилої риски й ідентифікує файл

(вершину), пошук якого ведеться від кореневої вершини дерева файлової системи з обходом тих галузок дерева файлів, що відповідають назвам окремих компонентів. Так, шляхи "/etc/passwd", "/bin/who" і "/usr/src/cmd/who.c" вказують на файли, що є вершинами дерева, зображеного на рис. 4, а шляхи "/bin/passwd" і "/usr/src/date.c" містять неправильний маршрут. Назва шляху пошуку не обов'язково має починатися з кореня, у ній варто вказувати маршрут щодо поточних для виконуваного процесу каталогу, при цьому попередні символи "похила риска" у назві шляху опускаються. Так, наприклад, якщо ми перебуваємо у каталозі "/dev", то шлях "tty01" вказує файл, повна назва шляху пошуку для якого "/dev/tty01".

Програми, виконувані під керуванням системи Linux, не містять ніякої інформації щодо внутрішнього формату, у якому ядро зберігає файли даних, дані в програмах представляються як безформатний потік байтів. Програми можуть інтерпретувати потік байтів за своїм бажанням, при цьому будь-яка інтерпретація ніяк не буде зв'язана з фактичним способом збереження даних в операційній системі. Так, синтаксичні правила, що визначають завдання методу доступу до даних у файлі, встановлюються системою і є єдиними для всіх програм, однак семантика даних визначається конкретною програмою. Наприклад, програма форматування тексту troff шукає наприкінці кожного рядка тексту символи переходу на новий рядок, а програма обліку системних ресурсів acctcom працює із записами фіксованої довжини. Обидві програми користуються тими самими системними засобами для здійснення доступу до даних у файлі як до потоку байтів, і всередині себе перетворять цей потік за відповідним форматом. Якщо кожна з програм знайде, що формат даних неправильний, вона вживає відповідних заходів.

Каталоги схожі на звичайні файли в одному відношенні; система представляє інформацію в каталозі набором байтів, але ця інформація містить в собі назви файлів у каталозі в оголошеному форматі для того, щоб операційна система і програми, такі як, наприклад, ls (виводить список назв і атрибутів файлів), могли їх виявити.

Права доступу до файлу регулюються встановленням спеціальних бітів дозволу доступу, зв'язаних з файлом. Встановлюючи біти дозволу доступу, можна незалежно керувати видачею дозволів на читання, запис і виконання для трьох категорій користувачів: власника файлу, групового користувача та інших. Користувачі можуть створювати файли, якщо дозволено доступ до каталогу. Знову створені файли стають листками в деревоподібній структурі файлової системи.

Для користувача система Linux трактує пристрої так, ніби вони були файлами. Пристрої, для яких призначені спеціальні файли пристроїв, стають вершинами в структурі файлової системи. Звертання програм до пристроїв має той же самий синтаксис, що і звертання до звичайних файлів; семантика

операцій читання і записи стосовно пристроїв у великому ступені збігається із семантикою операцій читання і запису звичайних файлів. Спосіб захисту пристроїв збігається зі способом захисту звичайних файлів: шляхом відповідного встановлення бітів дозволу доступу до файлів. Оскільки назви пристроїв виглядають так само, як і назви звичайних файлів, і оскільки над пристроями і над звичайними файлами виконуються ті самі операції, більшості програм немає необхідності розрізняти всередині себе типи оброблюваних файлів.

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ КОМАНД

1. Команда touch

touch *параметри дата файл(и)*

Встановлення поточної дати і часу як дати останнього доступу до файлу і дати останньої зміни файлу. Якщо спробувати застосувати команду **touch** до неіснуючого файлу, то вона створить новий файл.

Цінність цієї команди більша, ніж здається з першого погляду. Наприклад, деякі системи налаштовані так, що певні типи файлів видаляються, якщо до них тривалий час не здійснювався доступ. Команда **touch** допомагає вирішити цю проблему, дозволяючи легко обновити дату останнього доступу до файлу. Крім того, дата і час доступу до файлу або модифікування файлу можуть використовуватися такими командами, як **find** та **make**.

Команда **touch** використовує два формати дати і часу:

1) MMddhhmmyy, де MM – місяць (1…12), dd – число (1…31), hh – години (00…23), mm – хвилини (00…59), yy – рік (00…99).

2) yyMMddhhmm.

Параметри:

-a Встановлення лише дати і часу останнього доступу до файлу.

-c Заборона створення нового файлу, якщо файлу із зазначеною назвою не існує.

-m Встановлення лише дати і часу останнього модифікування файлу.

-t Встановлення дати і часу останнього доступу і модифікування. При цьому використовується другий формат дати і часу (yyMMddhhmm).

Увага! В Ubuntu працює другий формат дати і часу (yyMMddhhmm) і параметри **-at**, **-mt**, **-t**.

Споріднена команда:

date Встановлення системної дати і часу.

2. Команда stat

stat *файл*

Виведення інформації з дескриптора файлу, наприклад, режиму доступу до файлу, його часового штампа (дата і час доступу, дата і час модифікування,

дата і час зміни стану) тощо. У деяких версіях Linux ця команда відсутня, чи має назву **statx**.

3. Команда ls

ls *параметри каталог/файл*

Виведення списку файлів у каталозі. Якщо каталог не зазначений, використовується поточний каталог.

Ця команда є одночасно й однією з найпростіших (що може бути простіше, ніж виведення списку файлів у каталозі?), і однією з найскладніших (через велику кількість параметрів) команд Linux. Звичайно, не всі параметри цієї команди застосовуються однаково часто: найчастіше використовуються **-F** та **-l**, але навряд чи є сенс у використанні **-i** або **-c**.

Приклади:

```
$ls
data     figures      misc
newdata  personnel    expenses
financial
```

(Ця команда виводить список файлів і підкаталогів поточного каталогу).

```
$ls newdata
newdata
```

(Результат виконання цієї команди підтверджує, що файл **newdata** міститься у поточному каталозі).

```
$ls god
god not found
```

(Результат виконання цієї команди показує, що поточний каталог не містить зазначеного файлу).

```
$ls -a
.        ..           .mailrc
data     financials   misc
newdata  personnel
```

(Ця команда виводить список усіх файлів у поточному каталозі, включаючи сховані файли, назви яких починаються з крапки).

Параметри:

-1 Виведення інформації про кожен файл або каталог в окремому рядку.

-a Виведення списку усіх файлів і підкаталогів у каталозі, включаючи сховані файли.

-b Виведення символів, що не друкуються, як вісімкового коду.

-c Сортування за датою і часом останньої зміни інформації про статус файлу.

-C Виведення списку, відсортованого по стовпцях (значення за замовчуванням).
-d Виведення тільки назви підкаталогу, але не його змісту.
-f Розглядати *зразок* як назву каталогу, а не назву файлу.
-F Позначити файли, що виконуються, зірочкою (*), каталоги – похилою рискою (/) і символічні посилання – символом @.
-g Довгий формат виведення, без включення інформації про власника файлу.
-i Виведення номерів і-вузлів.
-l Довгий формат виведення.
-L Виведення інформації про вихідні файли замість символічних посилань.
-m Неформатоване виведення списку файлів і підкаталогів, розділених комами.
-n Те саме, що й **-l,** але замість ідентифікаторів користувача і групи виводяться відповідні номери.
-o Те саме, що й **-l,** але не виводиться інформація про групу, до якої належить файл чи каталог.
-p Виведення похилої риски наприкінці назви кожного каталогу.
-q Виведення символів, що не друкуються, як знаків питання (?).
-r Виведення у зворотному порядку.
-R Рекурсивна робота. Виводиться список усіх файлів поточного каталогу, списки усіх файлів у всіх його підкаталогах, списки усіх файлів у підкаталогах кожного підкаталогу поточного каталогу і т.д.
-s Виведення розміру файлів у блоках (за замовчуванням розмір виводиться у байтах).
-t Сортування за датою і часом останнього модифікування файлу. Файли, змінені останніми, будуть міститися на початку списку.
-u Сортування за датою і часом останнього доступу до файлу.
-x Виведення списку, відсортованого за рядками.

Споріднені команди:

chmod Зміна режиму доступу до файлу.
chgrp Зміна групи, до якого належить файл.
chown Зміна власника файлу.
find Пошук файлів.
ln Створення посилань.

4. Команда pwd

pwd

Команда виводить повну назву поточного каталогу.

Параметрів немає.

Споріднена команда:

cd Зміна поточного каталогу.

5. Команда cp

cp *параметри вихідний_файл кінцевий_файл*
cp *параметри файл каталог*
cp *параметри каталог1 каталог2*

Копіювання вмісту файлу у файл з іншою назвою або в інший каталог зі збереженням існуючої назви файлу, усіх файлів одного каталогу в інший каталог.

Приклади:

```
$cp kevin.memo kevin.memo.old
```

(Вміст файлу **kevin.memo** копіюється у новий файл з назвою **kevin. memo. old**).

```
$cp kevin.memo /usr/users/kevin/old_jnrk
```

(Файл **kevin.memo** копіюється у каталог **/usr/users/kevin/ old_junk**).

```
$cp -r /usr/users/kevin/old_junk /usr/users/kevin/backup
```

(Усі файли з каталогу **/usr/users/kevin/old_junk** копіюються у каталог **/usr/users/kevin/backup**).

Параметри:

-i Запит на підтвердження при перезаписі існуючих файлів.
-p Збереження існуючого режиму доступу до файлу. Цей параметр доступний не у всіх системах.
-r Копіювання цілого каталогу.

Споріднені команди:

chgrp Зміна групи, який належить файл.
chmod Зміна режиму доступу до файлу.
chown Зміна власника файлу.
ln Створення посилань.
mv Переміщення або перейменування файлу.
rm Видалення файлу.

6. Команда rm

rm *параметри файли*

Видалення файлів. Для того щоб мати право видалити файл, необхідно бути його власником, або мати дозвіл на запис у каталог, в якому міститься файл (недостатньо мати дозвіл на запис самого файлу). Якщо нема дозволу на запис файлу, але є дозвіл на запис у каталог, що його містить, перед видаленням файлу система запитає підтвердження. Команда **rm** також може використовуватися для видалення каталогів (оскільки каталог - це просто файл, що містить інформацію про інші файли).

УВАГА! Слід використовувати цю команду з обережністю. Після того, як файл видалений, його більше немає. Хоча існують деякі утиліти для відновлення видалених файлів (наприклад, програма *Norton Utilities* для деяких версій Linux), треба дуже акуратно поводитися з командою **rm**. Варто використовувати параметр **-i**, що вказує команді на необхідність запиту на підтвердження перед видаленням файлів.

Приклади:

```
$rm textfile
```

(Ця команда видаляє файл з назвою **textfile**).

```
$rm textfile?
```

(Ця команда видаляє усі файли, назва яких починається з **textfile,** і містить ще один символ, наприклад **textfilel, textfilea** і т.і.).

```
$rm -r stuff
```

(Ця команда видаляє каталог **stuff** і всі файли, що містяться в ньому, і підкаталоги).

Параметри:

-f Видалення файлів без запиту на підтвердження.
-i Обов'язковий запит на підтвердження при видаленні кожного файлу.
-r Рекурсивне видалення (видаляється даний каталог і всі файли, що містяться у ньому, і підкаталоги).

Споріднена команда:

rmdir Видалення каталогу.

7. Команда mv

mv *параметри вихідний_файл кінцевий_файл)*

Перейменування файлу або переміщення одного чи декількох файлів в інший каталог.

Приклади:

```
$mv 2006.report /users/home/misc
```

(Ця команда переміщає файл **2006.report** у каталог **/users/ home/misc**).

```
$mv 2006.report 2007.report
```

(Ця команда перейменовує файл **2006.report** у **2007. report**).

```
$mv 2006.report /users/home/misc/2007.report
```

(Ця команда переміщає файл **2006.report** у каталог **/users/ home/misc,** одночасно змінюючи назву файлу на **2007.report**).

```
$mv -i 2006.report /users/home/misc/2007.report
```

```
mv: overwrite 2007.report?
```
(Ця команда переміщає файл **2006.report** у каталог **/users/home/misc,** одночасно змінюючи назву файлу на **2007.report.** Оскільки зазначено параметр **-i** і файл **/users/home/misc/2007.report** вже існує, то до користувача є запит на підтвердження).

Параметри:

-f Заборона запиту на підтвердження при перезаписі вже існуючих файлів.

-i Вимога запиту на підтвердження при перезаписі існуючих файлів.

Споріднена команда:

cp Копіювання файлів.

8. Команда mkdir

mkdir *параметри каталоги*

Створення нового каталогу або декількох каталогів.

Приклади:

```
$mkdir stuff
```
(Ця команда створює в поточному каталозі підкаталог з назвою **stuff**).

```
$mkdir –m 444 stuff
```
(Ця команда створює в поточному каталозі підкаталог з назвою **stuff** і режимом доступу **444**).

Параметри:

-m *режим* Створення каталогу із заданим режимом доступу.

9. Команда cd

cd *каталог*

Робить зазначений каталог поточним робочим каталогом. Насправді це команда інтерпретатора командного рядка, але вона зазвичай вважається стандартною командою Linux.

Приклади:

```
$cd
```
(Робить ваш домашній каталог поточним каталогом).

```
$cd stuff
```
(Робить поточним каталогом підкаталог **stuff** поточного каталогу).

```
$cd /usr/users/eric/private
```
(Робить поточним каталогом каталог **/usr/users/eric/private**).

```
$cd -/stuff/2006
```
(Робить поточним каталогом підкаталог **stuff/2006** вашого домашнього каталогу).

```
$cd ..
```
(Робить поточним каталогом батьківський каталог поточного каталогу).

Параметрів немає.

Споріднена команда:

pwd Виведення назви поточного робочого каталогу.

10. Команда file

file *параметри файл*

Виведення типу заданого файлу. За необхідності команда **file** для визначення типу використовує файл **/etc/magic**.

УВАГА! Виведена інформація не завжди є правильною. Однак команда **file** добре працює при розпізнаванні текстових файлів, сценаріїв, файлів у форматі PostScript і команд Linux.

Параметри:

-c Використовувати інформацію з файлу **/etc/magic**.
-f *список* Запуск команди **file** для кожного файлу зі заданого списку.
-h Ігнорувати файли-посилання.
-m *файл* Використовувати зазначений файл замість файлу **/etc/magic**.

11. Команда chmod

chmod *параметри режим файли*

Ця команда змінює режим доступу до зазначеного файлу (файлів) чи вмісту усього каталогу. Тільки власник файлу чи привілейований користувач можуть змінювати режим доступу до файлу.

Існує два способи вказування нового режиму доступу: символьний і числовий. Числова форма використовується для встановлення абсолютного значення режиму доступу, а символьна – для зміни режиму доступу щодо поточного стану.

Щоб довідатися режим доступу до файлу чи каталогу слід використовувати команду **ls**.

Приклад використання числової форми:

```
$chmod 744 kevin.report
```

У цьому прикладі за допомогою команди **chmod** встановлено такий режим доступу до файлу, при якому власникові дозволяється читання, зміну і виконання даного файлу, тоді як іншим користувачам, незалежно від того, чи є вони членами групи, якій належить файл, дозволено тільки читання.

Значення **744** виходить при додаванні значень, що відповідають основним режимам, які перераховані нижче. Найменше можливе значення – **000**, яке означає, що ніхто не має права на читання, зміну або виконання файлу. Найбільше можливе значення – **777**, що надає всім користувачам права на читання, зміну і виконання файлу. Ось як отримано значення **714**:

400 Власник має право на читання файлу.
200 Власник має право на зміну файлу.
100 Власник має право на виконання файлу.
010 Члени групи мають право на виконання файлу.
004 Інші користувачі мають право на читання файлу.
714

Коли наступного разу запустити команду **ls,** то можна побачити, що файл **kevin.report** має такі дозволи на доступ:

```
rwx--xr--
```

Режими: це вісімкове число, отримане підсумовуванням потрібних доданків з такого списку:
400 Власник файлу має право на читання.
200 Власник має право на зміну файлу.
100 Власник має право на виконання файлу.
040 Члени групи, який належить файл, мають право на читання.
020 Члени групи мають право на зміну файлу.
010 Члени групи мають право на виконання файлу.
004 Всі інші користувачі мають право на читання файлу.
002 Всі інші користувачі мають право на зміну файлу.
001 Всі інші користувачі мають право на виконання файлу.

Наприклад, режим доступу **423** означає, що власник файлу має право на читання, користувачі групи мають право на зміну, всі інші мають право на зміну і виконання файлу. (Зауваження: звичайно, для виконання файлу потрібен дозвіл на читання).

Символьна форма:

Коли встановлення дозволів здійснюється таким способом, режими доступу вводяться в символьній формі, але структура команди залишається незмінною. Замість використання чисел для вказування режиму доступу слід використовувати такі символи:

u Користувач (власник файлу) – від *user* (*користувач*).
g Група – від *group* (*група*).
o Інші користувачі – від *other* (*інші*).
a Всі користувачі – від *all* (*усі*). Це значення за замовчуванням.
+ Додати дозволи до поточних.
- Видалити дозволи з поточних.
= Встановити дозволи незалежно від поточних.

r Дозвіл на читання – від *read* (*читати*).
w Дозвіл на запис – від *write* (*писати*).
x Дозвіл на виконання – від *execute* (*виконувати*).
l Блокування файлу для інших користувачів при доступі - від *lock* (*замок*).

Можна встановлювати більше одного режиму за раз, розділяючи параметри комами (ні ліворуч, ні праворуч від якої не має бути пробілу). Крім того, можна встановити за один раз права доступу для декількох категорій користувачів, як показано у прикладах, що наводяться нижче.

Приклади використання символьної форми:

```
$chmod u+x kevin.report
```

(Ця команда надає власникові файлу **kevin.report** право на виконання даного файлу).

```
$chmod u-x kevin.report
```

(Ця команда позбавляє власника файлу **kevin.report** права на виконання даного файлу).

```
$chmod u+x,go-w file.report
```

(Власник файлу **file.report** одержує право на його виконання, а члени групи, який належить файл, і всі інші користувачі позбавляються права на його зміну).

Параметр:

–R : Рекурсивна зміна режиму доступу. Змінюється режим доступу до каталогу і усіх файлів, що містяться у ньому, і підкаталогів.

Споріднені команди:

chgrp Зміна групи, якій належить файл.
chown Зміна власника файлу.
newgrp Зміна групи, до якої належить користувач.

12. Команда ln

ln *параметри вихідний_файл файл_посилання*
ln *параметри файли каталог*

Ця команда створює посилання на файл як прямі, так і символічні. Посилання на файли дозволяють здійснювати доступ до одного файлу за декількома назвами. Незалежно від кількості створених посилань на файл, на диску зберігається тільки один файл. Команда ln дозволяє також створювати посилання на файл з тою самою назвою, що має існуючий файл, але в іншому каталозі.

Найчастіше зручно використовувати *символічні посилання*, оскільки при використанні символічних посилань нема необхідності поміщати їх на файлову систему, що містить існуючий файл. Крім того, можна легко знайти

існуючий файл за символічним посиланням на нього, використовуючи команду **ls**.

УВАГА! Важливо не заплутатись у синтаксисі цієї команди. Випадково переплутавши порядок вказування назв, можна зіпсувати файли. Важливо запам'ятати: спочатку вказується назва існуючого файлу. Потім слід вказати назву посилання. Після виконання команди посилання буде вказувати на початковий файл.

Приклади:

```
$ln kevin eric
```

(Ця команда створює посилання з назвою **eric** на файл **kevin**).

```
$ln kevin /usr/users/kevin/misc
```

(Ця команда створює посилання з назвою **kevin** у каталозі **/usr/users/kevin/misc** на файл **kevin** у поточному каталозі).

Параметри:

-f Створити посилання, навіть якщо файл із зазначеною назвою вже існує.
-n Заборона зміни вже існуючих файлів.
-s Створення символічного посилання.

13. Команда mount

mount *параметри пристрій каталог*

Команда системного адміністрування. Монтування файлової системи. Команда **mount** сповіщає систему, що файлова система, яка монтується, присутня на *пристрої*. Точкою монтування є *каталог,* що має існувати і бути порожнім. Після монтування цей каталог стає коренем змонтованої файлової системи. Команда **mount** без аргументів перелічує всі підмонтовані файлові системи, їх точки монтування, відзначає системи, доступні тільки для читання, а також повідомляє дату монтування. Команда **mount** доступна тільки привілейованому користувачеві.

Параметри:

-a Монтувати усі файлові системи, перераховані у файлі */etc/fstab*.
Примітка: це єдиний параметр, що не сполучимо з аргументами *пристрій* і *вузол*.

-f Імітація монтування. Виконати всі перевірочні дії для пристрою і каталогу, але саме монтування не робити.

-n Не реєструвати монтування в */etc/fstab*.

-o *option*

Примітка: це єдиний параметр **mount,** що вимагає наявності аргументу *пристрій* або *вузол*. Аргумент *options* може приймати такі значення:

async Дозволити асинхронне введення/виведення для пристрою.
auto Дозволити монтування з параметром **-a**.

defaults Використовувати значення за замовчуванням для всіх режимів (**async**, **auto, dev, exec, nouser, rw, suid**).

dev Інтерпретувати всі спеціальні пристрої, що існують у системі.

exec Дозволити виконання двійкових файлів.

noauto Заборонити монтування з параметром **-a**.

nodev Не інтерпретувати спеціальні пристрої, що існують у системі.

noexec Заборонити виконання файлів.

nosuid Не брати до уваги біти **suid** і **sgid.**

nouser Дозволити доступ до файлової системи, яка монтується, тільки привілейованому користувачеві.

remount Вважати, що система вже підмонтована, і перемонтувати її.

ro Дозволити доступ тільки для читання.

rw Дозволити доступ для читання і запису.

suid Брати до уваги біти **suid i sgid**.

sync Синхронне введення/виведення для пристрою.

user Дозволити непривілейованим користувачам монтувати файлову систему. Слід звернути увагу, що значеннями за замовчуванням для такої системи будуть **nodev**, **поехес** і **nosuid,** якщо не зазначене інше.

check=relaxed | normal | strict Дозволяє вказати, наскільки строго має регулюватися інтеграція файлової системи **MS-DOS**.

conv=binary | text | auto Визначити метод перетворення файлів на файлових системах **MS-DOS** і **ISO 9660**.

debug Включити налагодження для файлових систем **MS-DOS** і **ext2fs.**

errors=continue | remount | ro | panic Дія, що виконується при виникненні помилки. Тільки для файлових систем **ext2fs**.

-r Монтування в режимі тільки для читання.

-t *type* Вказати тип файлової системи. Можливі значення: **minix, ext, ext2, xiafs, hpfs, msdos, umsdos, vfat, proc, nfs, iso9660, smbfs, ncpfs, affs, ufs, romfs, sysv**, **xenix** і **coherent.** Слід звернути увагу, що **ext** і **xiafs** доступно тільки для ядер версій менше 2.1.21 і що замість **Xenix** і **coherent** варто використовувати **sysv**.

-v Діагностика монтування.

-w Монтувати в режимі читання/запису. Режим за замовчуванням.

Файли:

/etc/fstab Список файлових систем, що монтуються і параметрів монтування.

/etc/mtab Список підмонтованих у даний момент систем і параметрів монтування.

14. Команда du

du *параметри файли каталоги*

Виведення величини простору на диску, зайнятого каталогом (з усіма його підкаталогами) у блоках (зазвичай 1 блок становить 512 чи 1K байтів – залежно від версії операційної системи). За замовчуванням виводиться інформація про поточний каталог.

Параметри:

-a Виведення інформації не тільки про каталоги, але й про файли.

-r Виведення інформації про файли і каталоги, що команда **du** не може відкрити.

-s Виведення тільки загального підсумку, без відображення проміжної інформації.

Споріднена команда:

df Виведення інформації про вільне місце на диску.

ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

1. Зареєструватись у системі.
2. Ознайомитися з призначенням та особливостями застосування рекомендованих команд.
3. Дослідити виконання команд з параметрами, які збігаються у методичних вказівках та на сторінках системного електронного посібника.
4. Скласти і захистити звіт.

ЗМІСТ ЗВІТУ

1. Номер, назва і мета лабораторної роботи.
2. Призначення і стислий опис команд з окремими параметрами.
3. Результати досліджень команд (запуск і реакція системи).
4. Висновки.

ЛАБОРАТОРНА РОБОТА № 4

Керування процесами та їх взаємодією в ОС Linux

МЕТА РОБОТИ: Ознайомитись з характеристиками процесів системи. Навчитись роботі з процесами.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Концепція процесів є базовою в ОС Linux. Саме ядро існує для забезпечення потреб процесів. Програма в термінах Linux є файлом, що виконується. Це основа побудови контексту процесу при його породженні. По суті породження будь-якого процесу в Linux - це створення деякої віртуальної машини. Вона має свій власний адресний простір, де містяться процедурний сегмент і сегмент даних.

Системні дані, використовувані ядром для ідентифікації процесу, що існують у плині всього часу життя процесу, утворюють дескриптор (опис) процесу. Безліч дескрипторів утворюють таблицю процесів. Розмір таблиці процесів, хоча і має припустимі обмеження, але в сучасних версіях Linux дозволяє створювати до декількох сотень процесів. Дескриптор процесу містить його параметри. Інформація про стан включає розташування (адреса в пам'яті), розмір частини образу процесу, що виводиться, ідентифікатори процесу і його користувача, що запустив.

Інша важлива інформація про процес зберігається в таблиці користувача (контексті процесу). Сюди записуються ідентифікаційні номери користувача і групи, щоб визначити привілеї доступу до файлів, посилання на системну таблицю файлів для усіх відкритих процесом файлів, покажчик на індексний дескриптор поточного каталогу в таблиці індексних дескрипторів і список реакцій на різні сигнали.

Процеси утворюють ієрархічні співвідношення між собою. Існують поняття "процес-батько" і "процес-син". Процеси можуть породжувати інші процеси, ці породжені процеси називають "процес-син". "Процес-син" успадковує усі властивості "процесу-батька", включаючи змінні оточення, поточну директорію, відкриті файли.

Процес, що породив, може зупинити своє виконання до завершення одного з "процесів-синів" за допомогою системного виклику wait(&status).

Значення, що буде повернуто в змінну status, містить у молодшому байті ідентифікатор процесу, який завершився, а в старшому байті – статус завершення. Будь-який процес може завершитися за власною ініціативою системним викликом exit(status). Аргумент status передається "процесові-батькові", якщо той очікує завершення "процесу-сина".

Операційна система Linux, будучи у своїй основі засобом керування процесами, сама по собі може розглядатися як система паралельних взаємодіючих процесів з деревоподібною структурою. Загальний прабатько

всіх процесів у системі - процес init, що перебуває у вершині генеалогічного дерева. Цей процес породжується особливим чином і далі постійно присутній у системі. Всі інші процеси породжуються в системі за уніфікованою схемою системним викликом fork().

Кожному створеному процесові Linux призначає унікальний ідентифікатор, відомий як процес ID (або абревіатура PID). Цей PID ідентифікує процес і для самої ОС, і для безлічі команд і системних викликів. На додаток до цьому кожен процес має також parent process ID (PPID), що є не що інше, як PID його "процесу-батька".

Використовуючи команду ps, можна завжди побачити ідентифікатори поточних процесів у системі. Наприклад, якщо назва облікового запису користувача terry, то після виконання команди ps -f на екрані можна побачити:

```
UID PID PPID C STIME TTY TIME COMMAND
terry 3865 3699 2 13:35:43 ttyp3 0:00 ps –f
terry 3699 3699 0 12:58:21 ttyp3 0:00 ksh
```

Цей фрагмент показує, що користувач terry має два процеси: команду ps -f і ksh (Korn Shell).

Процеси можуть функціонувати у двох режимах: системному і користувацькому. Робота в системному режимі означає виконання процесом системних викликів. Він найбільш важливий, тому що в ньому виконується обробка переривань, викликаних зовнішніми сигналами і системними викликами, а також керуванням доступом до диска, розподіл додаткової динамічної пам'яті та інших ресурсів системи. Процес функціонує в користувацькому режимі, коли виконується код, заданий користувачем.

Для кожного процесу створюється свій блок керування, що поміщається в системну таблицю процесів, які містяться у ядрі. Ця таблиця є масивом структур блоків керування процесами. В кожнім блоці утримуються такі дані:

– слово стану процесу,
– пріоритет,
– величина кванта часу, виділеного системним планувальником,
– ступінь використання системним процесором,
– ознака диспетчеризації,
– ідентифікатор користувача, якому належить процес,
– ефективний ідентифікатор користувача,
– реальний і ефективний ідентифікатори групи,
– група процесу,
– ідентифікатор процесу та ідентифікатор батьківського процесу,
– розмір образу, розташовуваного в області підкачування,
– розмір сегментів коду і даних,
– масив сигналів, що очікують на обробку.

Щоб система функціонувала належним чином, ядру необхідно відслідковувати всі ці дані.

Відразу після обробки системного виклику fork() відбувається встановлення двостороннього зв'язку між "процесом-батьком" і "процесом-

сином". "Син" при цьому успадковує усі властивості "батька". Однак існування нового, породженого процесу як точної копії іншого не має ніякого змісту. Тому майже завжди "син" заміняє програмний код "батька" на свій власний програмний код. Це можливо за допомогою системного виклику exec.

Кожний з процесів, що завершили свій життєвий шлях, має свого "батька", але одночасно він може бути "батьком" інших процесів. Тому можуть залишитися "сироти", долю яких треба якось вирішувати. Тут є декілька варіантів.

1. Варіант нормального закінчення. "Процес-батько" видає wait і чекає, коли "процес-син" завершить своє виконання по exit. Однак, якщо в "сина" свої "процеси-діти", щоб вони не залишилися "сиротами", відбувається їхнє "усиновлення" прабатьком усіх процесів init. Головна робота ядра Linux при цьому полягає у перевстановленні PPID в "процесів-синів". Він стає для них рівним 1. Крім цього, усім процесам – членам групи розсилається відповідний сигнал.

2. Варіант "ненормального" закінчення. "Процес-син" видає exit, коли його "батько" існує і не перебуває у стані очікування wait. У цьому випадку зв'язок "батька" з "сином" продовжує існувати, незважаючи на те, що "син" "помер". Слот "процесу-сина" у таблиці процесів зберігається і тому такий не до кінця знищений процес називають "зомбі".

3. Варіант передчасного виходу. "Процес-батько" закінчується раніше від своїх "синів". У цьому випадку відбувається реконфігурація генеалогічного дерева і всі "процеси-сироти" перенаправляються постійно діючому процесові init.

Якщо класифікувати процеси в Linux, то можна виділити користувацькі і системні процеси, а також процеси, що образно називаються "демонами". Більшість процесів мають статус користувацьких. Системні процеси виконуються в режимі суперкористувача та орієнтовані на виконання системних функцій.

Коли користувацькому процесові потрібно виконати системну функцію, він створює системний виклик. Фактично відбувається виклик ядра системи як підпрограми. З моменту появи системного виклику процес вважається системним. Таким чином, користувацький і системний процеси є двома фазами одного процесу, але вони ніколи не перетинаються між собою. Кожна фаза користується своїм власним стеком. Диспетчерський процес не має користувацької фази.

"Процеси-демони" - це визначений різновид системних процесів. "Демони" відрізняються від звичайних процесів тим, що вони виконують специфічні системні дії, наприклад, адміністрування і керування у мережах. Так, типовим „процесом-демоном” є оброблювач вхідних викликів на Linux-сервері, що обробляє всі заявки, які надходять, на з'єднання з віддаленого входу для машини в мережі. Способи породження і запуску "демонів" можуть бути різні:

1. У процесі старту системи з файлу /etc/rc. Такі "демони" будуть працювати у статусі суперкористувача під час роботи ОС.

2. Використовуючи або системний файл /usr/lib/crontab, або користувацький crontab. Зазвичай стандартний системний процес cron впродовж дня виконує визначені задачі, періодично вибираючи свої команди для таких виконань з файлу /usr/lib/crontab.

3. За допомогою команди at "демон" очікує настання заданого часу і видає визначене завдання.

4. Запуск фонового процесу за допомогою користувацького термінала.

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ КОМАНД І МЕТАСИМВОЛІВ

1. Команда ps

ps *параметри*

Виведення списку всіх запущених процесів. Коли ця команда використовується без параметрів, то виведений список містить інформацію про номер процесу PID, пов'язаний з ним термінал TTY, час роботи процесу TIME і командний рядок, за допомогою якого він був запущений COMMAND. Численні параметри дозволяють отримувати додаткову інформацію. У деяких версіях Linux параметри істотно відрізняються, наприклад, замість **ps -ef** необхідно використовувати **ps -aux**.

Параметри:

-a Виведення списку всіх процесів, за винятком провідних процесів груп і процесів, не зв'язаних з яким-небудь терміналом.

-c Виведення інформації про класи процесів, що використовуються планувальником задач.

-d Виведення інформації про всі процеси, крім провідних процесів груп.

-e Виведення інформації про усі без винятку процеси.

-f Виведення великої кількості інформації про процеси UID, PID, PPID, C, STIME, TTY, TIME і COMMAND.

-g *(список)* Виведення інформації про процеси, що належать групам, провідні процеси яких мають номери, зазначені у списку.

-j Виведення не тільки номера процесу, але і номерів його групи і сеансу.

-l Виведення великої кількості інформації про процеси, що включають, наприклад, пріоритети, встановлені командою **nice,** і багато чого іншого.

-o Виведення визначеної користувачем інформації про процеси.

-p *(список)* Виведення інформації про процеси, номери яких зазначені у списку.

-s *(список)* Виведення інформації про процеси, що належать сеансам, провідні процеси яких мають номера, зазначені у списку.

-t *(список)* Виведення інформації про процеси, зв'язані з одним із терміналів у списку.

-i *(список)* Виведення інформації про процеси, що належать одному з користувачів у списку.

Споріднені команди:

kill Переривання процесу.

nice Запуск процесу зі зниженим пріоритетом.

Функціональний еквівалент у командному рядку операційної системи Microsoft Windows: **tasklist**.

2. Команда top

top *параметри*

Команда виводить список процесів, відсортований за процесорним часом, який займають процеси.

Параметри:

-d *(секунди)* Вказування затримки між оновленнями списку.

-q Оновлення списку без затримок.

-S Виведення процесорного часу, що поглинається завершеними процесами-нащадками.

-s Заборона інтерактивного режиму.

-i Ігнорування процесів-зомбі та неактивних процесів.

3. Команда kill

kill *параметри PID*

Перериває процес, що виконується, посилаючи йому зазначений сигнал. Процес визначається своїм номером – PID, *Process Identificator*, що може бути отриманий за допомогою команди ps. Для виконання команди **kill** треба бути власником процесу або привілейованим користувачем. Ця команда також вбудована в інтерпретатори командного рядка Korn shell, Bourne shell і C shell, хоча між її версіями є незначні розходження.

kill -9 – найбільш характерна форма цієї команди.

Параметри:

-l Виведення списку всіх сигналів.

(-сигнал) Відправлення процесові зазначеного сигналу, заданого у числовій або символьній формі.

Споріднена команда:

ps Виведення списку процесів, що виконуються.

4. Команда wait

wait *номер_процесу*

Вказує інтерпретаторові командного рядка дочекатися закінчення виконання процесу із заданим номером, перш ніж запустити новий процес.

Параметрів немає.

Споріднені команди:

ps Виведення списку всіх працюючих процесів.
sleep Припинення виконання команд на зазначений час.

5. Команда nohup

nohup *команда аргументи &*
nohup *параметри*

Дозволяє команді виконуватися у фоновому режимі навіть після виходу користувача із системи.

Параметри:

-help Показати допомогу і завершити роботу.
-version Вивести інформацію про версії і завершити роботу.

6. Команда nice

nice *параметри команда аргументи*

Виконання зазначеної команди зі зниженим пріоритетом. Використовується для запуску програм, робота яких забирає значний час, без поглинання ними значної кількості ресурсів.

Параметр:

-n Величина, на яку зменшується пріоритет. За замовчуванням – 10.

7. Команда at

at *параметри час [дата] інкремент*
at *параметри [ідентифікатор]*

Команда **at** дає змогу виконати задані дії в зазначений час у зазначений день. Наприклад, можна роздрукувати кілька великих документів на принтері опівночі, щоб не займати принтер на кілька годин вдень, коли він може знадобитися іншим людям. При використанні команди at не знадобиться перебувати на роботі опівночі, щоб відправити документи на друк.

Команда at має два різних набори параметрів. Перший з них дозволяє помістити завдання у чергу і вказати час і дату, коли воно має бути виконано. Другий набір параметрів призначений для керування завданнями, що вже є в черзі.

Щоб помістити завдання у чергу потрібно після введення команди at з параметрами ввести команди, що мають бути виконані. Завершивши введення всього завдання, слід натиснути **Ctrl-D**. У зазначений час завдання буде виконано. Якщо завдання виведе яку-небудь інформацію на стандартний вивід, вона буде відправлена електронною поштою.

Примітка: Незважаючи на те що команда at призначена, у першу чергу, для використання системним адміністратором, вона також може використовуватися і звичайними користувачами. Однак користувачам може бути заборонений доступ до цієї команди. Якщо є повідомлення про помилку (at: you are not authorized to run at. Sorry), то слід звернутися до адміністратора системи.

Приклади:

```
$at llam
ls
<Ctrl+d>
```

(at зчитує завдання зі стандартного вводу. Необхідно ввести команди з клавіатури і завершити введення завдання, натиснувши **<Ctrl+d>**).

```
$at llam nov 1
$at llam nov 1, 2006
$at llam sun
$at now + 2 weeks
$at [параметри] [ідентифікатор]
```

Примітка: При поміщенні завдання у чергу його ідентифікатор виводиться системою на екран.

Параметри:

-f *(файл)* Читання завдання із зазначеного файлу, а не зі стандартного вводу. Цей параметр доступний не у всіх системах.

-m Повідомлення користувача після завершення виконання завдання.

(час) Час, за який має бути виконане завдання. Якщо явно не зазначене інше (за допомогою суфікса am або pm), то використовується 24-годинний формат часу.

midnight, noon, now Ці параметри позначають визначені моменти часу: **midnight** – північ, **noon** – полудень; якщо зазначено параметр **now** (зараз), то має бути зазначений *інкремент*.

(дата) Дата; як правило, використовується у такому форматі: місяць, день, рік. Аргумент *місяць* має бути стандартним трибуквенним скороченням англійської назви місяця (наприклад, Jan або Nov); аргумент *рік* може бути опущений.

(день) День тижня, у який має бути виконане завдання. Параметр *день* може бути або повною *(Sunday),* або скороченою *(Sun)* англійською назвою дня тижня.

today, tomorrow Ці параметри позначають визначені дні: **today** – сьогодні, **tomorrow** – завтра.

-l Виведення списку завдань у черзі.

-r Видалення завдання з черги.

(інкремент) Числове значення, що задає час виконання завдання щодо поточної дати і часу. Параметр і*нкремент* також має містити одну

з таких одиниць виміру: **minute** (хвилина), **hour** (година), **day** (день), **week** (тиждень), **month** (місяць), **year** (рік). У наведеному прикладі параметр **now + 2 weeks** означає, що завдання буде виконано рівно через два тижні після поміщення у чергу.

Споріднені команди:

atq Виведення списку задач, поставлених у чергу за допомогою команди at.

atrm Видалення задачі з черги.

batch Виконання послідовності команд у фоновому режимі.

8. Команда tee

tee *параметри файл*

Запис інформації на стандартному виводі у файл без перенаправлення стандартного виводу. Не використовуючи цю команду, неможливо записати стандартний вивід команди у файл *одночасно* з виведенням файлу на екран або передати його на введення двох різних команд. Команда **tee** ніколи не використовується сама по собі. Вона застосовується лише у складених командах.

Приклади:

```
$spell textfile |tee regularwords
```

(Команда **spell** буде виконана з аргументом **textfile,** причому інформація буде виведена на екран і одночасно записана у файл **regularwords**).

```
$ls |tee textfiie |wc
```

(Виведення команди **ls**, що є списком файлів у поточному каталозі, записується у файл **textfile** і одночасно передається на введення команди **wc**, яка підраховує кількість рядків, слів та символів).

Параметри:

-a Інформація дописується у кінець зазначеного файлу.

-i Ігнорувати системні переривання.

9. Метасимвол &

Будь-яке завдання в Linux можна запустити у фоновому режимі. Для цього треба у кінці командного рядка вказати символ **&**.

Приклад:

```
$ls -lar >ls.dat &
```

(Ця команда виводить повну інформацію про всі файли з поточного каталогу, а також всіх його підкаталогів у файл з назвою ls.dat. Оскільки це може зайняти чимало часу, то виконується у фоновому режимі).

10. Перенаправлення введення/виведення програм

Зазвичай команда читає вихідні дані з клавіатури (стандартне введення) і видає дані на екран (стандартне виведення). Часто буває, що треба це змінити. Наприклад, щоб відправити лист, який зберігається у якомусь файлі, треба набрати:

```
$mail -адреса_одержувача <назва_файлу_з_листом
```

Це дуже зручно. Якщо необхідно зберегти список файлів з якогось каталогу у файлі, треба набрати:

```
$ls параметри >назва_файлу_для_запису_списку_файлів
```

Метасимвол **>** (більше) змушує команду записувати дані замість стандартного виводу (екрана) у зазначений файл. Якщо файл існує, він буде знищений, після чого створений заново, але вже з виведеною інформацією.

Метасимвол **<** (менше) змушує команду читати вхідні дані не з клавіатури (стандартного вводу), а з файлу. Якщо файл не існує, буде видано повідомлення про помилку.

Метасимволи **>>** працюють аналогічно метасимволу **>**, але файл (якщо він існує) знищений не буде. Замість цього дані будуть дописані у кінець файлу.

Метасимвол **|** (вертикальна риска, “трубопровід”) змушує передати виведення однієї команди на обробку іншої. Саме так організовано поекранне виведення каталогу:

```
$ls -la |more
```

Це означає – все, що виведе команда **ls -la** буде передано як вхідні дані на обробку команді **more**.

Звичайно, можна вказувати кілька символів трубопроводу для запуску послідовності з багатьох команд, якщо це потрібно. Можна комбінувати будь-які метасимволи перенаправлення введення/виведення. Наприклад, якщо у файлі **la** лежить рядок **-la**, то вивести зміст каталогу у файл **ls.dat** можна так:

```
$ls <la >ls.dat
```

Тобто, взяти вихідні дані для команди **ls** з файлу **la** і записати результат її роботи у файл **ls.dat**.

Однак, не слід захоплюватися такими перенаправленнями. Останній приклад, зокрема, придатний для ілюстрації можливостей перенаправлення введення/виведення, але не для нормальної роботи.

За допомогою метасимволу **;** (крапка з комою) можна запустити на виконання послідовність команд. При цьому кожна наступна команда почне виконуватись лише після завершення попередньої.

ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

1. Зареєструватись у системі.
2. Ознайомитися з призначенням та особливостями застосування рекомендованих команд і метасимволів.
3. Дослідити виконання команд з параметрами, які збігаються у методичних вказівках та на сторінках системного електронного посібника.
4. Скласти і захистити звіт.

ЗМІСТ ЗВІТУ

1. Номер, назва і мета лабораторної роботи.
2. Призначення і стислий опис команд з окремими параметрами.
3. Результати досліджень команд і метасимволів (запуск і реакція системи).
4. Висновки.

ЛАБОРАТОРНА РОБОТА № 5

Системне адміністрування та робота з користувачами в ОС Linux

МЕТА РОБОТИ: Вивчити права доступу та способи захисту файлів. Навчитись роботі з користувачами в ОС Linux.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Вузли, приєднані до мережі – особливо до всесвітньої мережі Інтернет, – стикаються з більшою кількістю проблем, пов'язаних з безпекою, ніж інші. Система безпеки мережі скорочує ризик несанкціонованого під'єднання. Але за своєю природою доступ до мережі і комп'ютерна безпека мають прямо протилежні цілі. Мережа – це швидкісний шлях, що служить для транспортування даних та організації доступу до них комп'ютерних систем, тоді як зміст безпеки полягає у керуванні доступом до цих систем. Гарантування мережної безпеки – це процес пошуку рівноваги між відкритим доступом і безпечним доступом.

Одним з підходів до гарантування мережної безпеки є розподіл відповідальності за керування різними сегментами великої мережі і делегування її підрозділам організації. Такий підхід вимагає участі великої кількості людей та суперечить ідеології забезпечення безпеки шляхом централізації керування. Але розподіл відповідальності і керування між невеликими адміністративними групами дозволяє створювати середовище невеликих, простих для спостереження мереж з відомими групами користувачів.

При створенні підмережі слід призначити адміністратора. Адміністратор підмережі відповідає за її безпеку і призначення IP-адрес мережним пристроям, що під'єднуються. Призначення IP-адрес дозволяє адміністратору підмережі контролювати склад комп'ютерів, що під'єднуються через підмережу, а також допомагає адміністратору визначати, які комп'ютери входять у підмережу і хто відповідає за кожен з комп'ютерів. Коли адміністратор підмережі призначає комп'ютеру IP-адресу, він також делегує адміністратору цього комп'ютера певні обов'язки стосовно безпеки. Так само, коли адміністратор створює для користувача обліковий запис, користувачу ставиться за обов'язок дотримання певних норм безпеки.

Ієрархія відповідальності виглядає так: адміністратор мережі, адміністратори підмереж, адміністратори систем, користувачі. На кожному рівні ієрархії окремі особи наділяються певною відповідальністю і відповідними повноваженнями.

Коректне адміністрування допомагає забезпечити збереження продуктивності та безпеки серверів. Завдання полягає у розробленні і впровадженні таких політик безпеки, які найкраще підходять до конкретного мережного середовища.

В системах Linux користувачами можуть виступати як живі люди (облікові записи, прив'язані до фізичного користувача), так і логічні користувачі (облікові записи, призначені для того, щоб окремі прикладні програми могли виконувати певні задачі). Користувач, незалежно від того, до якого з цих типів він належить, має ідентифікатор користувача (User ID, UID – як правило, він є унікальним) та ідентифікатор групи (Group ID, GID). Для порівняння, групи є одиницею логічної організації в системі – вони зв'язують користувачів один з одним і надають їм права на читання, запис і виконання окремо взятого файлу. Кожен файл, що створюється, присвоюється певному користувачеві, певній групі. Для цього файлу призначається два комплекти повноважень на читання, запис та виконання: один – для власника файлу і групи, до якої цей файл прив'язаний, і другий – для решти користувачів даного комп'ютера.

Пізніше адміністратор (кореневий користувач) має право замінювати користувача і групу, пов'язаних з конкретним файлом (а також повноваження, задані відносно нього).

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ КОМАНД

1. Команда id

id *параметри користувач*

Команда **id** виводить інформацію про вказаного користувача (за замовчуванням – про користувача-ініціатора виконання команди). Виводиться системний ідентифікатор користувача і його номер, ідентифікатори і номери груп, до яких належить користувач, а також (якщо вони є) ефективний ідентифікатор користувача та ефективний ідентифікатор групи.

Параметри:

-a Виведення списку всіх груп, до яких належить користувач.
-g Виведення лише групи.
-G Виведення інформації про додаткові групи.
-n Виведення ідентифікаторів (а не номерів) при використанні параметрів **-g**, **-G**, та **-u**.
-r Виведення реальних, а не ефективних ідентифікаторів користувача і групи.
-u Виведення лише номера користувача.

Споріднені команди:

logname Виведення системного ідентифікатора користувача.
who Виведення списку користувачів.

2. Команда useradd

useradd *параметри name*

Команда **useradd** створює обліковий запис для нового користувача з назвою *name,* а також його початковий каталог, в який будуть скопійовані файли ініціалізації. Команда доступна лише системному адміністратору.

Параметри:

-u *uid* **[-o]** Визначає унікальне числове значення ідентифікатора користувача (UID) за винятком випадків використання параметра **[-o].**

-g *group* Визначає групу, до якої буде належати користувач після реєстрації в системі.

-G *group*[,…] Визначає членство користувача у додаткових групах, перелічених через коми у вигляді списку.

-d *name* Вказує початковий каталог користувача *name* у каталозі *home*.

3. Команда userdel

userdel [-r] *name*

Знищує обліковий запис користувача *name*. Команда доступна лише системному адміністратору.

Параметри:

-r Знищує також початковий каталог користувача разом з його вмістом.

Якщо користувач, обліковий запис якого передбачається знищити, працює в системі, така операція не буде виконана.

4. Команда usermod

usermod *параметри name*

Команда **usermod** змінює атрибути облікового запису користувача з назвою *name*, наприклад, домашній каталог. Команда доступна лише системному адміністратору.

Параметри:

-d *dir* Зробити вказаний каталог *dir* домашнім.

-l *new_name* Змінити назву облікового запису користувача *name* на назву *new_name*.

-g *new_group* Змінити первинну групу користувача *name* на групу *new_group*, яка вже має існувати до цього моменту.

-G *group*[,…] Визначає членство користувача у додаткових групах, перелічених через коми у вигляді списку.

-L Заблокувати обліковий запис користувача.

-U Розблокувати обліковий запис користувача.

5. Команда groupadd

groupadd *параметри groupname*

Створення нової групи з назвою *groupname*. Команда доступна лише системному адміністратору.

Параметри:

-g *gid* **[-o]** Створення групи з унікальним ідентифікатором *gid*, якщо не заданий параметр **[-o].** Параметр **[-o]** дозволяє дублювати GID у межах системи**.**

-r Додавання системного облікового запису і автоматичний вибір першого доступного GID, меншого ніж 499, якщо не заданий параметр **-g**.

-f Аварійне завершення команди, якщо група, що додається, вже існує в системі.

6. Команда chown

chown *параметри новий_власник файл(и)*

Зміна власника вказаного файлу. Новий власник може бути вказаний за допомогою його системного ідентифікатора чи номера.

Параметри:

-h Зміна власника посилання на файл.

-R Рекурсивназміна власника (тобто змінюється власник файлів в підкаталогах і файлів, на які вказують посилання).

Споріднені команди:

chmod Зміна режиму доступу до файлу чи каталогу.

chgrp Зміна групи, якій належить файл.

newgrp Зміна групи, до якої належить користувач.

7. Команда chgrp

chgrp *параметри група файл(и)*

Зміна групи, якій належить файл. Може бути вказана як існуюча група, так і група, що створюється наново. Група задається або назвою (яка зберігається у файлі **/etc/groups**), або числовим ідентифікатором. Звичайний користувач може виконати операцію зміни групи лише з тими файлами, власником яких він є. Адміністратор може виконати операцію зміни групи з будь-якими файлами. Команда також може бути використана для зміни групи, якій належить каталог і всі файли у ньому.

Параметри:

-h Зміна атрибутів символічних посилань, але не файлів, до яких відносяться посилання.

-R Рекурсивна зміна: змінюється група, якій належать всі файли і підкаталоги даного каталога.

група Зміна поточної групи на вказану. Аргумент *група* може бути як назвою, так і числовим ідентифікатором групи.

Споріднені команди:

chown Зміна власника файлу.
chmod Зміна режиму доступу до файлу чи каталогу.
newgrp Зміна групи, до якої належить користувач.

8. Команда w

w *параметри користувачі*

Виведення інформації про систему: списку користувачів, приєднаних в даний момент до системи, статистики використання системи, а також задач, що виконуються користувачами. Команда є комбінацією команд **who, ps –a** та **uptime**.

Інформація про систему виводиться в заголовку і включає: поточний час, час, що минув після останнього перезавантаження системи, кількість користувачів, що працюють в даний момент в системі, а також середнє завантаження системи за останні 1,5 та 15 хвилин.

Інформація, яка виводиться про користувачів, включає: системний ідентифікатор користувача, назву термінала, назву віддаленої системи, час роботи в системі, час неактивності, JCPU (час, що використовується всіма процесами даного термінала), PCPU (час, що використовується поточним процесом) та командний рядок поточного процесу.

Параметри:

-h Заборона виведення заголовка.
-u Заборона виведення інформації про PCPU та командні рядки..
-s Заборона виведення інформації про час роботи в системі JCPU і PCPU.
-f Виведення поля **from** (назва віддаленої системи).

Споріднені команди:

free, ps, top, uptime.

9. Команда df

df *параметри файлова_система*

Виведення інформації про величину вільного місця у файловій системі чи у файловій системі, зазначеній в параметрі *файлова_система*. Інші параметри дозволяють вивести величину вільного місця (у кілобайтах чи блоках) чи загальний обсяг диска.

Параметри:

-b Виведення величини вільного місця на диску в кілобайтах.
-e Виведення числа, що показує, скільки ще файлів може бути створено. Цей параметр використовується не у всіх системах.

-F *тип* Використовується для одержання інформації про незмонтовані файлові системи зазначеного *типу.* (Список використовуваних типів файлових систем у деяких версіях Linux може бути знайдений у файлі **/etc/vfstab**).

-g Повертає всю структуру **statvfs** для всіх незмонтованих файлових систем.

-k Виведення величини зайнятого місця в кілобайтах.

-l Виведення інформації лише про локальні файлові системи.

-n Виведення *типу* файлової системи. Цей параметр використовується не у всіх системах.

-t Виводить величину як вільного, так і зайнятого місця. Цей параметр використовується не у всіх системах.

Споріднена команда:

du Виведення інформації про використання місця на диску.

10. Команда groups

groups *користувач*

Виведення списку груп, членом яких є даний користувач. Якщо користувач не вказаний, то виводиться список груп, членом яких є користувач-ініціатор виконання команди.

Споріднені команди:

chgrp Зміна групи, якій належить файл чи каталог.

newgrp Зміна групи, до якої входить користувач.

11. Команда tail

tail *параметри файл*

Виведення декількох останніх рядків файлу. Зазвичай використовується адміністратором для операцій з log-файлами.

Параметри:

-f Очікування змін вмісту файлу і негайне відображення їх на екрані. Цей параметр часто використовується, коли файл збільшується у розмірі. Щоб завершити роботу команди, треба натиснути CTRL+C.

-r Виведення рядків у зворотньому порядку.

-*n*b Виведення останніх *n* блоків.

+*n*b Виведення всіх блоків після блоку *n.*

-*n*c Виведення останніх *n* символів.

+*n*c Виведення всіх символів після символа *n.*

-*n*l Виведення останніх *n* рядків.

+*n*l Виведення всіх рядків після рядка *n.*

Споріднена команда:

head Виведення декількох перших рядків файлу.

ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

1. Зареєструватись у системі.
2. Ознайомитися з призначенням та особливостями застосування рекомендованих команд.
3. Дослідити виконання команд з параметрами, які збігаються у методичних вказівках та на сторінках системного електронного посібника.
4. Скласти і захистити звіт.

ЗМІСТ ЗВІТУ

1. Номер, назва і мета лабораторної роботи.
2. Призначення і стислий опис команд з окремими параметрами.
3. Результати досліджень команд (запуск і реакція системи).
4. Висновки.

ЛАБОРАТОРНА РОБОТА № 6

Адміністрування мережі на ОС Linux

МЕТА РОБОТИ: Ознайомитись з командами для адміністрування мережі на ОС Linux.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Задачі мережного адміністрування діляться на дві різні категорії: налаштування та усунення несправностей. Задачі налаштування зв'язані з передбачуваними явищами: вони вимагають детальних знань у синтаксисі команд, але прості та прямолінійні. Коли система коректно налаштована, рідко виникає причина змінювати настройки. Процес налаштування повторюється кожного разу після встановлення нової версії операційної системи, але з мінімальними варіаціями.

І навпаки, діагностування мережних проблем пов'язано з непередбачуваними явищами. Мережні проблеми зазвичай неповторні. Вирішення проблем – важлива складова частина забезпечення стабільності та надійності роботи мережі.

Однією з найскладніших задач, пов'язаних з технічним супроводом мереж, є виявлення відмінностей між правильною та неправильною поведінкою мережі. Зазвичай мережі є складними зі заплутаною структурою середовищами, у яких спрогнозувати запити кожної робочої станції (а також визначити реакцію мережі у кожному можливому випадку) майже неможливо. Отже, пошук несправностей у мережах нерідко ґрунтується на порівнянні поточних експлуатаційних характеристиках з еталонним рівнем продуктивності (Baseline, базисом). Саме цей параметр визначає, що є "нормальним" для даної мережі. Якщо рівень продуктивності мережі опускається нижче встановлених базисів, адміністратор може провести кількісне вимірювання змін, проаналізувати їх, і навіть використати ці дані як аргумент для проведення модернізації.

Мережі об'єднують комп'ютери таким чином, щоб різні системи могли обмінюватися інформацією. Системи Linux традиційно надають користувачам і адміністраторам набір простих, але дуже зручних мережних сервісів, що дозволяють перевірити стан системи через мережу, переглянути файли, які є на іншому вузлі, спілкуватися за допомогою електронної пошти тощо.

Щоб забезпечити роботу більшості програм у мережі, на кожній системі має безупинно працювати обслуговуючий фоновий процес, що реагує на запити клієнтів. Такий процес називається демоном.

Велика частина мережних команд Linux ґрунтується на протоколах Інтернету (Internet Protocols, IP). Протоколи є стандартизованими способами обміну інформацією через мережу і розбиті на ієрархічні шари. Дії протоколів простягаються від адресації і маршрутизації пакетів (на відносно низьких

рівнях) до пошуку користувачів і виконання команд користувача (на відносно вищих рівнях).

Основні користувацькі команди, які підтримуються більшістю систем через інтернет-протоколи, зазвичай називаються командами TCP/IP (назва, успадкована від двох найбільш розповсюджених мережних протоколів). Всі ці команди можна використовувати для зв'язку з будь-якими Linux-системами. Багато команд придатні також і для роботи з неUnix-системами, оскільки протоколи TCP/IP підтримуються багатьма системи.

ОСОБЛИВОСТІ ЗАСТОСУВАННЯ КОМАНД ТА УТИЛІТ

1. Команда telnet

telnet *параметри система*

Дозволяє встановити з’єднання з віддаленою системою за допомогою протоколу TELNET.

Параметри:

-a Спроба автоматичної реєстрації на віддаленій системі.

2. Команда hostname

hostname *параметри система*

Виведення назви локальної системи.

Параметри:

-d Виведення назви сервера DNS.
-f Виведення повної назви системи.
-s Виведення короткої назви системи.

Приклад:

```
$hostname -f
```

3. Команда host

host *параметри система сервер*

Виведення IP-адреси вказаної системи, використовуючи службу DNS. Якщо вказати IP-адресу, вона буде перетворена у назву системи.

Параметри:

-a Пошук записів типу ANY.
-A Пошук IP-адреси для вказаної системи, а потім назви системи за знайденою адресою і перевірка збігу знайденої назви з початковою. Перевірка IP-адрес всіх вузлів тієї самої зони. Якщо всі назви збігаються, не виводиться нічого.
-c Пошук записів ресурсів вказаного класу (ANY, CH, CHAOS, CS, CSNET, HS, HESIOD, IN, INTERNET, *). За замовчуванням – IN.
-C Виведення списку всіх систем у зоні (домені без піддоменів).

-d Режим налагодження.
-D Виведення кількості окремих систем у зоні, а також назв систем, які мають більше однієї адреси.
-f *файл* Перенаправлення стандартного виводу у вказаний файл.
-F *файл* Перенаправлення стандартного виводу у вказаний файл і відправлення на стандартний вивід додаткової інформації.
-l *зона* Виведення інформації про всі системи у вказаній зоні (домені без піддоменів).
-r Використання інформації з кешу сервера.
-R Пошук систем у доменах за замовчуванням.

Приклад 1:

```
$host -d svm-070 |less
```

Приклад 2:

```
$host –l eom.cs.lp.lviv.ua |less
```

4. Команда users

users *файл*

Виведення списку приєднаних до системи користувачів. Для отримання цієї інформації використовується файл **/var/run/utmp**.

Споріднена команда:

who

Приклад:

```
$users /var/run/utmp
```

5. Команда ping

ping *параметри система*

Команда відправляє ICMP-пакети ECHO_REQUEST на вказану систему для визначення пропускної здатності мережі.

Параметри:

-f Відправлення пакетів з тією ж швидкістю, з якою вони повертаються.
-i *n* Пауза *n* секунд перед відправленням пакетів.
-I *n* Відправлення *n* пакетів з максимальною швидкістю, потім повернення у звичайний режим роботи.
-p *зразок* Вказування байтів для заповнення пакета.
-q Виведення лише початкового і кінцевого повідомлення.
-R Встановлення параметра RECORD_ROUTE для пакетів, що відправляються і виведення отриманої з принятих пакетів інформації про шлях.

-s *n* Встановлення кількості байт даних в пакеті. За замовчуванням – 56.

-v Виведення додаткової інформації.

Приклад:

```
$ping 172.16.1.115
```

Для переривання слід натиснути комбінацію клавіш <Ctrl+c>. Можна використати адреси з діапазону від 172.16.1.11 до 172.16.1.164.

Функціональний еквівалент у командному рядку операційної системи Microsoft Windows: **ping**.

6. Команда traceroute

traceroute *параметри система*

Виведення мережного шляху від локальної системи до віддаленої і часу, який необхідний пакетам для проходження цього шляху. Якщо віддалена система не відповідає впродовж 5 секунд, виводиться зірочка. Тоді для переривання слід натиснути комбінацію клавіш <Ctrl+c>.

Приклад:

```
$traceroute 172.16.1.115
```

Функціональний еквівалент у командному рядку операційної системи Microsoft Windows: **tracert**.

7. Утиліта nslookup

nslookup *параметри система сервер*

Опитування служби імен DNS. Утиліта може використовуватись в одному з двох режимів: командному чи діалоговому. Для запуску у командному режимі першим аргументом вказується назва або IP-адреса шуканої системи (приклад 1). Діалоговий режим ініціюється запуском утиліти без аргументів (приклад 2). Вихід з діалогового режиму здійснюється інструкцією **exit**.

Інструкції діалогового режиму:

help Виведення переліку інструкцій діалогового режиму.

система Виведення інформації про *систему*.

система сервер Виведення інформації про *систему*, використовуючи вказаний *сервер*.

finger *користувач* Виведення інформації про *користувача*.

ls *домен* Виведення списку адрес *домену*.

exit Вихід з діалогового режиму.

Приклад 1:

```
$nslookup svm-070
```

Приклад 2:

```
$nslookup
```

8. Утиліта ifconfig

ifconfig *параметри*

Конфігурування мережного інтерфейсу. Без параметрів відображає стан поточних активних мережних інтерфейсів. Запускається з каталога **sbin**. Для переривання перегляду слід натиснути клавішу **<q>**.

Приклад:

```
$ifconfig |less
```

В результаті виконання утиліти на екран буде виведена така інформація:

Inet addr	IP-адреса.
Bcast	Адреса для широкомовного розсилання повідомлень.
Mask	Маска підмережі.
UP	Інтерфейс активний.
MTU	Максимальний розмір блоку передавання.
Metric	Ціна посилання пакета вказаним інтерфейсом.
RX packets	Кількість отриманих пакетів.
TX packets	Кількість переданих пакетів.
Errors	Кількість помилкових пакетів.
Dropped	Кількість відкинутих пакетів.
Overruns	Кількість неопрацьованих пакетів.
Frame	Кількість кадрів (блоків даних).
Carrier	Кількість носіїв.
Collisions	Кількість конфліктів.
Txqueuelen	Довжина черги передавання.
Interrupt	Кількість переривань.
Base addr	Кількість базових адрес.

Функціональний еквівалент у командному рядку операційної системи Microsoft Windows: **ipconfig**.

9. Утиліта route

route *параметри*

Програма прокладання статичних маршрутів через інтерфейси, налаштовані та активізовані програмою **ifconfig**. Без параметрів (чи з параметром -n) – виведення поточної таблиці маршрутизації протоколу IP. Запускається з каталога **sbin**.

Приклад:

```
$route
```

В результаті виконання утиліти на екран буде виведена така інформація:

Destination IP-адреса кінцевого вузла маршруту.
Gateway Назва або IP-адреса шлюзу, що використовується маршрутом (якщо шлюз не вказаний, відображається символ *).
Genmask Маска мережі для маршруту.
Flags Ознаки маршруту (U –маршрут активний, H – вузол, G – шлюз, D – динамічна маршрутизація, M – змінений) .
Metrik Ціна шляху.
Ref Кількість інших маршрутів, що стосуються даного.
Use Скільки разів використовувався даний елемент таблиці маршрутизації.
Ifase Мережний інтерфейс, через який проходить маршрут.

10. Команда netstat

netstat *параметри*

Перевірка стану мережі. Без параметрів – список активних мережних з'єднань. Для переривання перегляду слід натиснути клавішу **<q>**.

Параметри:

-a Виведення інформації про всі з'єднання.
-i Виведення статистики про всі мережні пристрої (як утиліта **ifconfig**).
-c Виведення інформації про поточний стан мережі з періодичністю в одну секунду (до переривання).
-n Виведення інформації про віддалені і локальні адреси та порти.
-o Виведення інформації про стан таймерів та додаткової інформації.
-r Виведення таблиці маршрутизації (як утиліта **route**).
-t Виведення інформації лише про TCP-сокети.
-u Виведення інформації лише про UDP-сокети.
-v Версія програми **netstat**.
-x Виведення інформації про доменні сокети Linux.

Приклад 1:

```
$netstat -a |less
```

В результаті виконання команди на екран будуть виведені дві таблиці з інформацією про:

1. Активні приєднання до Internet, де

Proto Протокол (TCP чи UDP) .
Recv-Q Кількість отриманих сокетом байт, що не скопійовані програмою користувача.
Send-Q Кількість байт, відправлених, але не розпізнаних віддаленим вузлом.

Local Adress Назва і порт локального вузла з'єднання (якщо не вказано -n, то IP-адреса транслюється у канонічну назву, а назва порту – у назву сервісу) .

Foreign Address Назва віддаленого вузла з'єднання (якщо не вказано -n, то IP-адреса транслюється у канонічну назву, а назва порту – у назву сервісу).

State Поточний стан сокета:

ESTABLISHED – з'єднання встановлено;
SYN_SENT – спроба встановлення з'єднання сокета з віддаленим вузлом;
SYN_RECV – з'єднання встановлюється;
FIN_WAIT1 – сокет закрився і чекає завершення з'єднання;
FIN_WAIT2 – з'єднання закрите, очікується закриття з іншої сторони;
TIME_WAIT – сокет закрився та очікує від віддаленого сокета припинення передачі;
CLOSED – сокет не використовується;
CLOSE_WAIT – віддалений сокет закрив з'єднання, локальний сокет очікує закриття з'єднання;
LOAST_ACK – з'єднання перервалось і сокет закрився, локальний сокет очікує підтвердження;
LISTEN – сокет прослуховує мережу для спроби встановлення нового з'єднання;
UNKNOWN – невідомий стан сокета;
USER – назва облікового запису користувача, що володіє сокетом.

2. Активні доменні сокети Linux, де

Proto Протокол, що використовується сокетом.

RefCnt Кількість процесів, підключених до сокета.

Flags Ознаки.

Type Режим доступу до сокета:

OCK_DGARM – режим дейтаграми, без з'єднання;
OCK_STREAM – режим потоку зі з'єднанням;
OCK_RAW – режим, що не обробляється;
OCK_RDM – режим надійної доставки інформації;
OCK_SEQPACKET – режим послідовних посилань;
UNKNOWN – невідомий режим.

State Поточний стан сокета:

FREE – сокет не розподілений;
LISTENING – сокет очікує запитів на встановлення з'єднання;
UNCONNECTED – сокет намагається встановити з'єднання;
CONNECTED – сокет встановив з'єднання;

DISCONNECTION – сокет намагається розірвати з'єднання;
UNKNOWN – стан невідомий.

Path Шлях, що використовується іншим процесом для під'єднання до сокета.

Приклад 2:

```
$netstat -i |less
```

В результаті виконання команди на екран буде виведена таблиця з такою інформацією:

Ifase Назва мережного інтерфейсу.

Відсортовані параметри мережі (порівняти з результатом виконання утиліти **ifconfig**).

Flags Ознаки:

A – інтерфейс отримує багатоадресні пакети;
B – інтерфейс отримує пакети широкомовного розсилання;
D – увімкнений режим налагодження;
L – інтерфейс зворотньої петлі;
M – інтерфейс перебуває у змішаному режимі;
N – інтерфейс не обробляє трейлери в пакеті;
O – для цього інтерфейсу відключений протокол ARP;
P – інтерфейс використовується для канального з'єднання типу "point-to-point";
R – інтерфейс запущений;
U – інтерфейс активізований.

Приклад 3:

```
$netstat -o |less
```

В кінці кожного рядка таблиці "Активні приєднання до Internet" додається інформація про увімкнення/вимкнення таймера; у дужках – час, що залишився до таймауту та кількість спроб.

Функціональний еквівалент у командному рядку операційної системи Microsoft Windows: **netstat**.

11. Команда talk

talk *користувач термінал*

Ведення інтерактивного діалогу двома користувачами. При виконанні команди екран розбивається навпіл. Текст, який вводиться користувачем, буде з'являтися у верхній половині екрану, а повідомлення іншого користувача – у нижній. Для завершення роботи слід натиснути комбінацію клавіш <Ctrl+c>.

Приклад:

Спочатку командою **who** чи **w** необхідно визначити, з яким користувачем і терміналом можна зв’язатись. Потім командою **talk** задати назву облікового запису користувача і номер термінала:

```
$talk ksm-5 ttyp2
```

У відповідь користувач, який отримав виклик, мусить підтвердити готовність до діалогу командою **talk**. У ній вказати назву облікового запису користувача, що був ініціатором зв’язку, і номер його термінала:

```
$talk ki-4 ttypb
```

12. Команда write

write *користувач* [*термінал*]

Відправлення користувачу повідомлення чи відповіді на його репліку. Сеанс роботи завершується введенням символу **EOF** (комбінація клавіш <**Ctrl+d**>). Якщо користувач працює більш, ніж на одному терміналі, слід вказати номер **tty**. Порядок встановлення інтерактивного діалогу подібний до команди **talk**.

ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

1. Зареєструватись у системі.
2. Ознайомитися з призначенням і особливостями застосування рекомендованих команд та утиліт.
3. Дослідити виконання команд з параметрами, які збігаються у методичних вказівках та на сторінках системного електронного посібника.
4. Скласти і захистити звіт.

ЗМІСТ ЗВІТУ

1. Номер, назва і мета лабораторної роботи.
2. Призначення і стислий опис команд та утиліт з окремими параметрами.
3. Результати досліджень команд та утиліт (запуск і реакція системи).
4. Висновки.

СПИСОК ЛІТЕРАТУРИ

1. Робачевский А.М. Операционная система UNIX. – СПб.: BHV – Санкт-Петербург, 2005. – 528 с.
2. Хант К. TCP/IP. Сетевое администрирование. – СПб.: Символ-Плюс, 2004. – 816 с.
3. Немет Э., Снайдер Г., Хейн Т. Руководство администратора Linux. – М.: Вильямс, 2003. – 880 с.
4. Сивер Э., Спейнауэр С., Фиггинс С., Хекман Д. Linux. Справочник. – 3-е изд. – СПб.: Символ-плюс, 2001. – 912 с.
5. Скловская С. Л. Команды Linux. Справочник. – СПб.: ДиаСофт ЮП, 2004. – 848 с.
6. Інтерактивна система перегляду системних посібників. [Електронний ресурс]. – Режим доступу: http://www.opennet.ru/man.shtml.

НАВЧАЛЬНЕ ВИДАННЯ

АДМІНІСТРУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ

для студентів першого (бакалаврського) рівня вищої освіти
спеціальності 123 "Комп'ютерна інженерія"

Укладач Хомуляк Мирослав Олегович

Редактор

Комп'ютерне верстання

Підписано до друку . .2020.
Формат 60x84 1/16. Папір офсетний. Друк на різографі.
Умовн. друк. арк. . Обл.-вид. арк. .
Наклад прим. Зам. .

Видавництво Національного університету "Львівська політехніка"
Реєстраційне свідоцтво ДК № 751 від 27.12.2001 р.

Поліграфічний центр Видавництва
Національного університету "Львівська політехніка"

вул.Ф. Колесси, 2, Львів, 79000