Міністерство освіти та науки України
Тернопільський національний технічний університет
імені Івана Пулюя

Кафедра
комп'ютерно-інтегрованих
технологій

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт

з дисципліни
"Операційні системи"

Тернопіль 2022

УДК 004.42
ББК 32.973

Методичні вказівки до виконання лабораторних робіт з дисципліни “Операційні системи” (для студентів напрямку підготовки 151 Автоматизація та комп’ютерно-інтегровані технології)

Розробили

доцент, к.т.н. Микитишин А.Г.
доцент, к.т.н. Чихіра І.В.

Рецензент доцент, к.т.н. Коноваленко І.В.

Методичні вказівки розглянуті на засіданні кафедри.
Протокол № 1 від 29 серпня 2022р.

Схвалено і рекомендовано до друку науково-методичною комісією факультету прикладних інформаційних технологій та електроінженерії Тернопільськтго національного технічного університету ім.Івана Пулюя
Протокол № 1 від 30 серпня 2022р.

Лабороторна робота №1

Тема: Навігація по файловій системі, базові команди роботи з файлами та каталогами.
Мета: Навчитись переходити по каталогах файлової системи та переглядати вміст каталогів та текуче розміщення в файловій системі, переіменування, перенос, копіювання файлів та каталогів. Створення та видалення файлів та каталогів.

1.2 Файлова структура

Файли в операційній системі Linux організовані в ієрархічну систему каталогів. Каталог може містити файли й інші каталоги. Через подібність з деревом таку структуру часто називають деревоподібною структурою. Будь-який каталог є підкаталогом іншого каталогу. Кожен каталог може містити безліч підкаталогів, але сам повинен бути підкаталогом лише одного каталогу.

Файлова структура ОС Linux розгалужується на кілька каталогів, починаючи з кореневого /. У кореневому каталозі є кілька системних каталогів, що містять файли і програми, які відносяться до самої ОС Linux. Кореневий каталог містить також каталог home, у якому можуть зберігатися початкові каталоги всіх користувачів системи. Початковий каталог кожного користувача, у свою чергу, буде містити в собі каталоги, які користувач створює для своїх потреб. Кожний з цих каталогів теж може містити каталоги. Усі ці вкладені каталоги відгалужуються від початкового каталогу користувача, як показано на мал. 1.

1.3 Початкові каталоги

При реєстрації в системі як робочий приймається початковий каталог користувача, ім'я якого зазвичай збігається з реєстраційним ім'ям. Робочий каталог можна змінити за допомогою команди **cd**. У процесі зміни робочого каталогу користувач переходить з одного каталогу в інший.

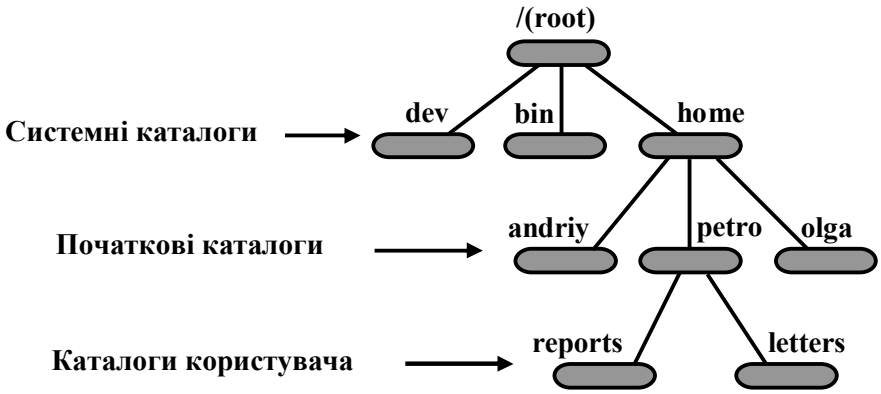

Рис. 1. Файлова структура Linux

1.6 Командний інтерпретатор

Командний інтерпретатор являє собою стрічковий інтерактивний інтерфейс між користувачем і операційною системою. Користувач вводить у командному рядку команди, командний інтерпретатор інтерпретує їх і посилає у виді інструкцій в операційну систему для виконання. Здатність інтерпретувати команди наділяє командний інтерпретатор багатьма корисними можливостями. Наприклад, у командному інтерпретаторі є набір групових символів, що дозволяють вибирати імена файлів з використанням зразків.

1.7 Базові операції виводу файлів: ls, cat, more і lpr

Одна з основних функцій операційної системи — керування файлами. Користувач постійно виконує з файлами базові операції виводу: відображення на екрані, друк й ін. У системі Linux використовується набір команд, що дозволяють виконувати базові операції керування файлами – перегляд списку файлів, відображення на екрані, друк, копіювання, перейменування і видалення. Ці команди, як правило, мають вид скорочених форм слів. Наприклад, команда **ls** — це скорочення від list; вона дозволяє переглянути список файлів, що містяться у каталозі. Команда **lpr** — скорочення від line print; вона забезпечує друк файлу і т.д.

Для одержання списку імен файлів і каталогів, які зберігаються в поточному каталозі використовується команда **ls**. Вона має багато опцій, що дозволяють одержувати спеціальним чином організовані списки імен файлів. При використанні даної команди без опцій вона видає список усіх файлів та каталогів, які містяться в поточному каталозі.

```
$ ls
weather reports letters
```

Для того щоб імена файлів і імена каталогів можна було б відрізнити один від одного, команду **ls** потрібно використати з опцією **–F**.

```
$ ls -F
weather reports/ letters/
```

Для виводу на екран всіх файлів поточного каталогу, включаючи також сховані файли, потрібно використати опцію **–a**.

```
$ ls -a
```

В якості параметра команда може використовувати ім’я чи шляхове ім’я каталогу чи файлу.

```
$ ls reports
mondey
```

Для отримання детальної інформації про файли і каталоги можна скористатись опцією **–l** команди **ls**.

```
$ ls -l mydata
-rw-rw-r-- 1 petro weather 207 Feb 20 11:55 mydata
```

Спочатку відображаються права доступу, за ними кількість посилань, ім'я власника файлу, ім'я групи, в яку входить користувач, розмір файлу в байтах, дата і час останнього відновлення файлу і його ім'я. Права доступу вказують, хто може мати доступ до файлу: користувачі, члени групи чи всі інші користувачі. Права доступу докладно розглянемо нижче. З прикладу видно, що файл mydata відноситься до числа звичайних файлів. Кількість посилань дорівнює 1; це значить, що файл не має інших імен і на нього немає інших посилань. Ім'я власника petro збігається з реєстраційним ім'ям, а ім'я групи буде weather. Розмір файлу — 207 байтів. Файл обновлявся останній раз 20-го лютого, у 11:55. Ім'я файлу — mydata.

Для отримання детальної інформації лише про каталог потрібно використати попередню команду з опцією **–d** команди.

```
$ ls -l reports
drwxr-x--- 2 petro 512 Feb 10 10:50 reports
```

Першим символом в полі прав є **d**, що вказує на те, що даний об’єкт є каталогом.

У багатьох випадках буває необхідно переглядати уміст файлу. Команди **cat** і **more** виводять вміст файлу на екран. Ці команди викликаються з ім'ям файлу, який потрібно переглянути:

```
$ cat mydata
computers
```

Команда **cat** виводить на екран відразу весь текст файлу. Якщо файл великий, то текст дуже швидко миготить на екрані. Для усунення цього недоліку служить команда **more**, за допомогою якої текст на екран можна виводити порціями.

```
$ more mydata
```

Коли **more** викликає файл, то відображається його перший фрагмент, що уміщається на екрані. Для відображення наступного фрагмента потрібно натиснути клавішу [f] чи клавішу пробілу. Для повернення до попереднього тексту використовується клавіша [b]. Натиснувши клавішу [q], можна в будь-який момент вийти з даної програми.

Для друку файлу, потрібно переслати його на принтер за допомогою команд **lpr**. У наступному прикладі користувач дає команду друкувати файл **mydata**.

```
$ lpr mydata
```

Якщо потрібно відразу надрукувати кілька файлів, можна вказати їх імена в командному рядку. У

наступному прикладі користувачу необхідно надрукувати файли **mydata** і **preface**:

```
$ lpr mydata preface
```

Завдання на друк ставляться в чергу і виконуються у фоновому режимі. Команда **lpq** дозволяє в будь-який момент перевірити хід виконання завдань на друк. З її допомогою на екран виводяться ім'я власника завдання (реєстраційне ім'я користувача, що послав це завдання), ідентифікатор завдання, розмір у байтах і

ім'я тимчасового файлу, у якому воно у даний момент знаходиться. У нашому прикладі власник – petro, а ідентифікатор завдання - 00015:

```
$ lpq
Owner ID    Chars  Filename
petro 00015 360    /usr/lpd/cfa00015
```

Для знаття будь-якого завдання з друку потрібно скористатися командою **lprm**. В якості параметра ця команда використовує ідентифікаційний номер завдання чи ім'я його власника. Вона дозволяє видалити завдання з черги на друк. Перед її виконанням дуже корисно ввести команду **lpq**, так як вона дозволяє з'ясувати номер і ім'я власника завдання на друк, які необхідно використовувати в команді **lprm.** У наступному прикладі скасовується завдання на друк:

```
$ lprm 00015
```

1.3 Керування каталогами: mkdir, rmdir, cd і pwd

Користувач може створювати і видаляти власні каталоги, а також змінювати свій робочий каталог. Для створення каталогів використовують команду **mkdir**. В якості параметра можна використовувати шляхові імена каталогів. У наступному прикладі користувач спочатку створює каталог reports, а потім, використовуючи повне шляхове ім'я — каталог letters.

```
$ mkdir reports
$ mkdir /home/petro/letters
```

Для видалення каталогу потрібно застосувати команду **rmdir** з ім'ям цього каталогу. У приведеному нижче прикладі користувач спочатку видаляє командою **rmdir** каталог **reports**, а потім, указавши повне шляхове ім'я — каталог **letters**.

```
$ rmdir reports
$ rmdir /home/chris/letters
```

У кожному каталозі можна створювати інші каталоги. Команда **cd** дозволяє переходити з одного каталогу в інший. Для того щоб визначити, у якому каталозі знаходиться користувач використовується команда **pwd**, яка повідомляє повне шляхове ім'я робочого каталогу.

```
$ pwd
/home/petro
```

Перехід у інший каталог робить його робочим. Коли користувач входить в систему, робочим каталогом стає його початковий каталог. У ході створення облікового запису користувача система створює початковий каталог для нового користувача. При реєстрації система завжди переводить користувача в його початковий каталог. Команда **cd** дозволяє зробити робочим інший каталог. Як параметр команда **cd** використовує ім'я каталогу, у який потрібно перейти. В якості імені каталогу може бути зазначене ім'я підкаталогу робочого каталогу чи повне шляхове ім'я будь-якого каталогу в системі. Для переходу в початковий каталог досить увести команду **cd** без параметрів.

```
$ pwd
/home/petro
$ cd reports
$ pwd
/home/petro/reports
$ cd /home/olga
$ pwd
/home/olga
```

Для позначення батьківського каталогу використовується спеціальний символ (дві крапки ..). Даний символ допускається використовувати в команді **cd** для переходу назад у батьківський каталог, таким чином знову роблячи цей каталог поточним. У наступному прикладі користувач переходить у каталог **letters**, а потім повертається в початковий каталог.

```
$ cd letters
$ pwd
/home/petro/letters
$ cd ..
$ pwd
/home/petro
```

Каталог завжди має батьківський каталог (крім кореневого каталогу). Зокрема, у попередньому прикладі каталог **petro** є батьківським для каталогу **letters**. При створенні каталогу в ньому відразу ж створюються два записи. Один з них буде представлений крапкою (.), а другий — двома крапками (..). Крапка позначає шляхове ім'я даного каталогу, а дві крапки — шляхове ім'я його батьківського каталогу.

```
$ cd letters
$ ls .
thankyou
$ ls ..
weather letters
```

1.8 Операції з файлами і каталогами: cp, mv, rm

Щоб створити копію файлу, потрібно вказати команді **cp** в якості параметрів два імені файлу. Перше з них — ім'я копіюючого файлу, друге — ім'я, яке потрібно привласнити копії. Це буде новий файл, що містить копію всіх даних вихідного файлу. Команда **cp** має наступний синтаксис:

```
$ вихідний_файл файл_результат
```

У наступному прикладі користувач копіює файл **text1** у новий файл **text2**:

```
$ cp text1 text2
```

Для того щоб скопіювати файл із робочого каталогу в інший, потрібно вказати ім'я цього каталогу команді **cp** як другий параметр. Нова копія буде мати те ж ім'я, що й оригінал, але розміститься в іншому каталозі.

```
$ cp імена_файлів ім'я_каталогу
```

Команда **cp** може використовувати в якості параметрів імена багатьох файлів, задані у виді списку, тому можна одночасно копіювати в каталог відразу кілька файлів. У наступному прикладі користувач копіює файли **preface** і **doc1** у каталог **props**.

```
$ cp preface doc1 props
```

У командному інтерпретаторі використовується ряд спеціальних символів, названих груповими символами, за допомогою яких можна вибирати групи файлів. Цими символами є зірочка, знак питання і квадратні дужки (*, ?, []).

Символ зірочка (*) – встановлює відповідність з будь-яким набором символів в іменах файлів. Нехай, наприклад, потрібно скопіювати в заданий каталог усі файли з вихідними текстами програм, написаними мовою С. Замість того щоб вказувати в командному рядку всі ці файли, можна ввести груповий символ * з розширенням с (*.c) У наступному прикладі користувач копіює усі файли вихідних текстів програм на С з поточного каталогу в каталог sourcec.

```
$ cp *.c sourcec
```

У наступному прикладі користувач копіює усі файли з каталогу **props** у каталог **oldprop**.

```
$  cp props/* oldprop
```

Знак питання (?) – позначає тільки один не вказаний символ в іменах файлів. Нехай, потрібно скопіювати файли docl і docA, але не файл document у каталог **newdoc**. У той час як зірочка позначає імена файлів будь-якої довжини, знак питання обмежує співставлення тільки одним символом. У наступному прикладі копіюються файли, які починаються з комбінації “doc”, після якої йде одна буква.

```
$ ls
doc1 docA document
$ cp doc? newdoc
```

Квадратні дужки [ ] – дозволяють задавати набір припустимих символів для пошуку. Система буде шукати в імені файлу кожної з цих символів. Нехай, потрібно скопіювати файли, що починаються з комбінації "**doc**", але завершуються тільки символами 1 чи A в каталог **newdoc**. Імена, що закінчуються символами 2, B и т. д., вам не потрібні. Нижче показано, як це зробити:

```
$ ls
doc1  doc2  doc3  docA docB  docD  document
$ cp doc[lA] newdoc
```

При копіюванні файлу можна дати копії ім'я, відмінне від імені оригіналу. Для цього потрібно помістити нове ім'я файлу після косої риски, що слідує за ім'ям каталогу.

```
$  cp ім'я_файлу ім'я_каталогу/нове_ім'я_файлу
```

У наступному прикладі файл **newprop** копіюється в каталог **props** і копії привласнюється ім'я **version1**. Потім користувач переходить у каталог **props** і одержує список файлів. У ньому є тільки один файл, що називається **version1**.

```
$  cp newprop props/version1
$  cd props
$  ls
version1
```

Система Linux дозволяє копіювати цілі каталоги. Перший параметр команди **cp** – ім'я файлу, який копіюється, а другий — ім'я каталогу, у який він буде поміщений. Для копіювання каталогу команду **cp** необхідно використовувати з опцією **–r**. Ця опція дає команді cp вказівку копіювати каталог разом із усіма його підкаталогами. Іншими словами, копіюється все дерево каталогів, починаючи з зазначеного. У наступному прикладі каталог thankyou копіюється в каталог oldletters. Після завершення цієї операції починають рівноправно співіснувати два підкаталоги thankyou: один у каталозі letters, інший в oldletters.

```
$ cp -r letters/thankyou oldletters
$ ls -F letters
/thankyou
$ ls -F oldletters
/thankyou
```

За допомогою команди **mv** можна або перейменувати файл, або перемістити файл з одного каталогу в інший. Використовуючи mv для перейменування файлу, в якості другого параметру потрібно вказати нове ім'я файлу. Перший параметр — поточне ім'я файлу.

```
$ mv поточне_ім'я_файлу  нове__ім'я_файлу
```

У наступному прикладі ім'я файлу **text** міняється на **version1**.

```
$ mv text version1
```

Файл можна перенести з одного каталогу в інший. Для цього потрібно в якості другого параметра в команді **mv** поставити ім'я каталогу. У даному випадку команда **mv** не перейменовує файл, а просто переміщає його з одного каталогу в інший. Після переміщення файлу в нього залишиться те ім'я, яке він мав у вихідному каталозі.

```
$ mv ім'я_файлу ім'я_каталогу
```

Якщо потрібно перейменувати файл одночасно з його переміщенням, то для цього вкажіть нове ім'я файлу після імені каталогу. Ім'я каталогу і нове ім'я файлу повинні бути розділені косою рисою. У наступному прикладі файл newprop перемет переміститься з початкового каталогу в каталог props і одержує ім'я version1.

```
$ mv newpropa props/version1
$ cd props
$ ls
version1
```

При створенні списку імен файлів для команди **mv** можна використовувати будь-які групові символи, описані вище.

Для видалення файлів використовується команда **rm**. У наступному прикладі користувач видаляє файл **oldprop**.

```
$ rm oldprop
```

Команда **rm** може бути використана з будь-яким числом параметрів, що дозволяє одночасно видаляти кілька файлів. Імена цих файлів вказуються в командному рядку після імені команди.

```
$ rm proposal version1 version2
```

Для видалення каталогу і всіх його підкаталогів застосовується команда **rm** з опцією **–r**. У наступному прикладі показано, як видалити каталог reports і всі його підкаталоги:

```
$ rm -r reports
```

Засоби та підготовка:
Логін та пароль до робочого місця з компютером, що працює під ОС Linux.

2.Порядок виконання роботи:
Запустити систему та залогуватись.
Показати текуче розміщення в файловій системі.
Ввести команду pwd
Написати шлях (повинен бути вашим домашнім каталогом).
1.Перегляд вмісту каталогів:
а) переглянути текучий каталог.
б) переглянути кореневий каталог.
в) переглянути каталог на два кроки вверх по дереву каталогів.
г) переглянути домашній каталог.
2.Переміщення по дереву каталогів:
а) перейти в кореневий каталог. Перевірка командою pwd.
б) перейти в домашній каталог по абсолютному шляху. Перевірка командою pwd.
в) перейти на два рівні вверх використовуючи спеціальні символи. Перевірка командою pwd.
г) перейти в домашній каталог по реляційному шляху. Перевірка командою pwd.
д) перейти в каталог /etc по абсолютному шляху. Перевірка командою pwd.
е) перейти в домашній каталог по спецйльному символу. Перевірка командою pwd.
є) перейти в каталог /etc по реляційному шляху. Перевірка командою pwd.
ж) вернутись в домашній каталог. Перевірка командою pwd.
3.Створення файлів та каталогів:
а) створити в домашньому каталозі новий каталог з назвою lab. Перевірка командою ls.
б) перейти в створений каталог. Перевірка командою pwd.
в) створити в даному каталозі файл newfile. Перевірка командою ls.
3.Копіювання, перенос та переіменування файлів та каталогів:
а) скопіювати каталог labs і тим самим створити новий каталог labsnew з їх вмістом. Перевірка командою ls.
б) переіменувати в каталозі labsnew файл newfile в oldfile. Перевірка командою ls.
в) скопіювати з каталогу labs файл newfile в каталог labsnew. Перевірка командою ls.
4.Видалення файлів та каталогів.
а) видалити файл newfile з каталогу labs. Перевірка командою ls.
б) видалити порожній каталог labsnew. Перевірка командою ls.
в) видалити весь каталог (з його вмістом) labs. Перевірка командою ls.
5.Завершити роботу системи.

3. Порядок оформлення звіту по роботі.

3.1 На титульній сторінці вказати назву університету, кафедри, назву і номер роботи, прізвище, ініціали, номер групи виконавця, прізвище і ініціали викладача, який керував роботою, рік виконання роботи.

3.2 Вказати мету роботи.
3.3 Привести завдання для розробки.

4. Контрольні запитання.

1. Яка команда призначена для перегляду текучого розміщення в файловій системі?
2. Якою командою можна переглянути кореневий каталог?
3. Якою командою можна перейти на каталог вищого рівня.
4. Яка команда призначена для створення каталогу?
5. Якою командою перейменовують файл і каталог?
6. Якою командою можна витерти файл і каталог?
7. Як скопіювати каталог під іншим іменем?

Лабороторна робота №2

Тема: Зміна прав доступу до файлів та каталогів
Мета: Визначати права доступу до файлів та каталогів та змінювати їх.

1. Підготовчі відомості:

1.1. Права доступу до файлів і каталогів: команда chmod

Для кожного файлу і каталогу в ОС Linux задаються права доступу, що визначають, хто і які операції може проводити над даним файлом. Користувач може керувати типом прав доступу до певного файлу чи каталогу. Для файлу чи каталогу може бути встановлене право на читання, запис і виконання. При створенні файлу автоматично встановлюються дозволи на його читання і запис для власника. Власник може змінювати права доступу як завгодно. Право тільки на читання запобігає внесенню у файл змін. Для файлу можна задати також право на виконання, що дозволяє виконувати його як програму.

Існує три категорії користувачів, що можуть мати доступ до файлу чи каталогу: власник, група та інші. Власник — це користувач, що створив файл. Часто користувачі поєднуються в групи. Наприклад, системний адміністратор може об'єднати в групу користувачів, які працюють над одним проектом. На свій розсуд користувач може дозволити чи не дозволити доступ до файлу членам своєї групи. Нарешті, користувач може відкрити доступ для всіх інших користувачів системи. У цьому випадку доступ до файлу чи каталогу будуть мати усі, хто працює у системі.

Для кожної категорії користувачів існує окремий набір прав доступу на читання, запис і виконання. Перший набір керує доступом самого користувача до його файлів — це права власника. Другий набір керує доступом групи до файлів цього користувача. Третій набір керує доступом інших користувачів до файлів даного користувача. Ці три набори прав доступу на читання, запис і виконання для трьох категорій — власника, групи й інших — утворять у сукупності дев'ять типів дозволів на дії з файлом.

Команда **ls** з опцією **-l** видає докладну інформацію про файл, включаючи права доступу до нього. У наступному прикладі перший набір символів ліворуч — це список прав доступу, установлених для файлу mydata

```
$  ls -l mydata
-rw-r--r-- 1 petro weather 207 Feb 20 11:55 mydata
```

Відсутність права доступу позначається дефісом (-). Право на читання позначається буквою **r**, право на запис — буквою **w**, право на виконання — буквою **x**. В першій групі символів десять позицій. Перший символ позначає тип файлу. Якщо перший символ — дефіс, то мова йде про файл. Якщо це буква **d**, видається інформація про каталог. Наступні дев'ять символів указують на права доступу для різних категорій користувачів. Перший трисимвольний набір — це права доступу до файлу для категорії “власник”. Другий трисимвольний набір — це права доступу до файлу для категорії “група”. Третій трисимвольний набір — це права доступу до файлу для категорії “інші”. З прикладу видно, що для файлу **mydata** встановлене право на читання і запис по категорії “власник”, право на читання по категорії “група” і право на читання по категорії “інші”. Це значить, що читати файл можуть усі члени групи і всі користувачі системи, а змінювати — тільки власник.

Для створення різних конфігурацій прав доступу застосовується команда **chmod**. В якості параметрів в цій команді використовуються два списки: список змін прав доступу і список імен файлів. Список змін прав доступу можна задавати двома способами: символічним і абсолютним. При використанні першого застосовуються символи **r, w, x**, а при використанні другого – використовується двійкова маска.

1.2. Символьний метод встановлення прав доступу

При символьному методі право на читання позначається буквою **r**, право на запис — буквою **w**, право на виконання — буквою **x**. Кожний з цих прав доступу можна додавати і видаляти. Символом додавання права доступу є знак плюс +. Символом скасування є знак мінус —. У наступному прикладі команда **chmod** додає право на виконання і скасовує право на запис для файлу **mydata**. Право на читання не змінюється.

```
$ chmod +x-w mydata
```

Використовуються й інші символи, що позначають категорії користувачів. Категорія “власник”, “група” і “інші” позначаються відповідно символами **u, g** і **o**. Символ категорії ставиться перед символами, що встановлюють права на читання, запис і виконання. Якщо символу категорії немає, то мають на увазі всі категорії, і зазначені права встановлюються для користувача, групи і інших. У наступному прикладі перша команда **chmod** встановлює право на читання і запис для групи. Друга команда **chmod** встановлює право на

читання для інших користувачів. Зверніть увагу на те, що пробілів між специфікаціями прав доступу і категорією немає. Список змін прав доступу являє собою одну довгу фразу без пробілів.

```
$ chmod g+rw mydata
$ chmod o+r mydata
```

Користувач може не тільки встановлювати, але і скасовувати права доступу. У наступному прикладі, для інших користувачів встановлюється право на читання, а права на запис і виконання скасовуються.

```
$ chmod o+r-wx mydata
```

Є ще один символ дозволу, **a** (від слова all), що позначає всі категорії. Він діє по замовчуванню. У наступному прикладі обидві команди еквівалентні.

```
$ chmod a+r mydata
$ chmod +r mydata
```

Крім прав на читання, запис і виконання можна встановлювати для користувачів право володіння програмами на час їх виконання. Під час виконання програми її власником є той користувач, що її запустив, навіть якщо сам файл програми належить іншому користувачу. Установка біта "зміна ідентифікатора користувача" дозволяє багатьом користувачам виконувати програму, користаючись при цьому правами її дійсного власника. Наприклад, багатьма програмами в системі володіє користувач root, тоді як виконують їх звичайні користувачі. Іноді при роботі таких програм виникає необхідність зміни файлів, що належать користувачу root. У цьому випадку звичайному користувачу потрібно запустити цю програму зі збереженням прав користувача root, щоб вона одержала можливість змінювати файли, що належать привілейованому користувачу. Біт зміни ідентифікатора групи має те ж значення, що і біт зміни ідентифікатора користувача, але тільки для груп. Користувачі виконують програму з правами тієї групи, до якої належить власник даної програми. Така програма може вносити зміни у файли, що належать групі.

Для установки біта зміни ідентифікатора користувача чи групи використовується опція **s**. У приведеному нижче прикладі біт зміни ідентифікатора користувача встановлюється в програмі **pppd**, якою володіє користувач root. Коли звичайний користувач запускає цю програму, вона виконується з правами її власника, і, таким чином, стає можливою зміна належних йому файлів.

```
$ chmod +s /usr/sbin/pppd
```

Біт зміни ідентифікатора користувача чи групи позначається буквою **s** у стовпці "виконання" категорії "власник" чи "група". Це право фактично є варіантом права виконання, **x**. Право на читання, право на запис і право володіння, таким чином, позначаються як **rws**, а не як **rwx**.

```
$ ls -l /usr/sbin/pppd
-rwsr-sr-x 1 root  root  84604  Feb  14  2003 /usr/sbin/PPPd
```

1.3 Абсолютні права доступу: двійкові маски

Абсолютний метод дозволяє змінювати відразу усі права доступу. Тут використовується двійкова маска, що позначає всі права доступу в кожній категорії. Три категорії з трьома правами доступу в кожній представлені у вісімковому форматі. При перетворенні у двійковий формат кожен вісімковий розряд перетворюється в три двійкових. Три вісімкових розряди числа перетворяться в три набори по три двійкових розряди в кожнім. Разом виходить дев'ять двійкових цифр, що в точності відповідає кількості прав доступу до файлу.

Вісімкові цифри можна використовувати як маску для встановлення різних прав доступу до файлу. Кожна вісімкова цифра відноситься до однієї з категорій користувачів. При цьому категорії нумеруються зліва на право, починаючи з категорії "власник". Перша вісімкова цифра відноситься до власника, друга - до групи, а третя — до інших користувачів.

Обрані користувачем вісімкові цифри будуть визначати права доступу на читання, запис і виконання по кожній категорії. Можна вважати, що вісімкова цифра спочатку перетвориться в двійкову форму, а потім групи двійкових розрядів використовуються для встановлення прав на читання, запис і виконання. Кожен двійковий розряд визначає одну зі складових прав доступу. Якщо двійковий розряд - 0, дозвіл відсутній. Якщо двійковий розряд - 1, дозвіл встановлений. Перший зліва двійковий розряд встановлює і знімає право на читання, другий — право на запис, третій - право на виконання. Наприклад, вісімкова цифра 6 перетвориться в двійкові цифри 110. Це відповідає встановленню права на читання і запис і скасовує право на виконання.

При використанні двійкової маски потрібно вказувати відразу три цифри, які забезпечують встановлення прав доступу одночасно для всіх трьох категорій. Це робить двійкову маску менш гнучкою, ніж

символи прав доступу. Щоб встановити право на виконання і скасувати право на запис для власника файлу mydata із збереженням права на читання, необхідно використовувати вісімкову цифру 5 (101 у двійковому форматі). Також потрібно вказати відповідні цифри для групи й інших користувачів. Якщо право на читання для цих категорій зберігається, то для кожної з них варто указати вісімкову цифру 4 (100). Це дає три вісімкові цифри 544.

```
$  chmod 544 mydata
```

Найчастіше двійкова маска використовується при встановленні права на виконання. Такі файли називаються сценаріями командного інтерпретатора. Для того щоб команди сценарію були виконані, потрібно спочатку вказати, що цей файл — виконавчий, тобто містить команди, які система може виконати. Це можна зробити різними способами, один із яких — установлення для сценарію права на виконання. Вісімкова цифра 7 (111) встановлює всі три дозволи, включаючи право на виконання (її можна одержати і додаванням цифр 4 (читання), 2 (запис) і 1 (виконання)). Цифра 0 для групи й інших користувачів забороняє їм усі види доступу. У наступному прикладі права доступу для власника файлу myprog включають право на виконання:

```
$ chmod 700 myprog
```

Щоб позначити біт зміни ідентифікатора користувача і групи, потрібно ввести перед цими вісімковими цифрами ще одну. Біт зміни ідентифікатора користувача позначається цифрою 4 (100); біт зміни ідентифікатора групи позначається цифрою 2 (010). У приведеному нижче прикладі для програми pppd установлюється біт зміни ідентифікатора користувача, а також права доступу на читання і виконання по всім трьох категоріях.

```
$ chmod 4555 /usr/sbin/pppd
```

1.4. Права доступу до каталогів

Права доступу можна встановлювати і для каталогів. Якщо для каталогу встановлене право на читання, користувачі можуть одержувати список файлів, що знаходяться в даному каталозі. Право на виконання дозволяє переходити в цей каталог. Право на запис дає можливість користувачу створювати і видаляти файли в цьому каталозі. Якщо користувач надасть іншим користувачам право на запис у своєму каталозі, вони зможуть додавати в нього файли. При створенні каталогу для нього автоматично встановлюються права на читання, запис і виконання для власника.

Як і у випадку з файлами, права доступу до каталогів задаються для власника, групи й інших користувачів. Часто виникає необхідність надати іншим користувачам можливість переходити в один з каталогів і одержувати список файлів у цьому каталозі, але не додавати туди свої файли. У даному випадку потрібно установити для каталогу право на читання і виконання, а право на запис не встановлювати. Це дозволить іншим користувачам переходити в даний каталог і одержувати список файлів, що знаходяться в ньому, але не створювати в ньому нові файли. У наступному прикладі для каталогу **newdir** встановлюється право на читання і виконання по категорії “група”, а право на запис скасовується. Члени групи можуть входити в каталог **newdir** і одержувати список його файлів, але створювати в ньому нові файли вони не можуть.

```
$ chmod g+rx-w letters/newdir
```

У вісімковому вигляді

```
$ chmod 750 letters/newdir
```

Засоби та підготовка:
Логін та пароль до робочого місця з компютером, що працює під ОС Linux.

2.Порядок виконання роботи:
2.1Запустити систему та залогуватись.
2.2 Створити в домашньому каталозі каталоги labs та labs1 і у них файли file1 та file2.
2.3 Представити права доступу до створених каталогів та файлів та провести їх опис в символьному та восьмирковому вигляді.
2.4 Змінювати прада доступу більш підходящим методом (символічним чи восьмирковим):
Таблиця перетворень в восьмирковий вид

	r	***w***	***x***
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0

	r	*w*	*x*
5	1	0	1
6	1	1	0
7	1	1	1

а) дозволити всім користувачам системи заходити у свій домашній каталог. Перевірка командою ls -l.
б) дозволити користувачам своєї групи повний доступ до каталогу labs з підкаталогами та заборонити любий доступ анонімним користувачам. Перевірка командою ls -l.
в) виставити повний доступ собі та своїй групі та тільки на читання всім користувачам на каталог labs1 з вмістом. Перевірка командою ls -l.
г) виставити повний доступ на свій домашній каталог з падкаталогами тільки для себе і закрити для всіх решта. Перевірка командою ls –l.
Завершити роботу системи.

3. Порядок оформлення звіту по роботі.

3.1 На титульній сторінці вказати назву університету, кафедри, назву і номер роботи, прізвище, ініціали, номер групи виконавця, прізвище і ініціали викладача, який керував роботою, рік виконання роботи.
3.2 Вказати мету роботи.
3.3 Привести завдання для розробки.

4. Контрольні запитання.

4.1 Яка командою можна змінити права доступу до файлу?
4.2 Якою командою можна дати доступ до файла усім користувачам?
4.3 Якою командою можна дати доступ до файла користувачам певної групи?
4.4 Які права буде на файли при восьмирковому коді 744?

Лабороторна робота №3

Тема: Робота з текстовим редактором vi.
Мета: Робота з типовим консольним повноекранним текстовим редактором Linux.

1. Підготовчі відомості:

Редактор **vi** (назва є скороченням від “visual”, візуальний) є одним з широко розповсюджених редакторів в ОС Linux. В даному редакторі клавіатура використовується для виконання двох зовсім різних функцій: для вводу команд редагування і для вводу символів самого тексту.

Деякі команди редагування, такі як **a** чи **i**, переводять клавіатуру в режим вводу. Якщо, знаходячись у командному режимі, користувач натисне клавішу [**i**], то перейде в режим вводу, у якому клавіші будуть представляти символи тексту. При натисканні клавіші [Esc] відбувається повернення в командний режим. Під час редагування тексту приходиться часто переходити з одного режиму в інший. У табл. 1 приведений набір основних команд, необхідних для роботи з редактором vi .

Щоб почати редагування файлу, у рядку командного інтерпретатора потрібно набрати vi і ім'я файлу. Якщо такий файл не існує, система створить його. По суті, присвоєння імені файлу, якого не існує, дає vi вказівку створити його. Приведена нижче команда викликає редактор vi з файлом test. Якщо цього файлу немає, він буде створений.

```
$ vi test
```

Після виконання команди vi редактор переходить у командний режим. Кожна клавіша починає виконувати функції команди редагування, а екран стає вікном з текстовим файлом. Текст виводиться на екран посторінково. Спочатку виводиться перша сторінка тексту і курсор встановлюється в лівий верхній кут. Якщо буде редагуватися тільки що створений файл, то зрозуміло, що ніякого тексту на екрані не буде. Цей факт відбивається стовпцем тильд з лівої сторони екрана. Тильди позначають частину екрана, яка не є частиною файлу.

Оскільки, після запуску редактор vi знаходиться в командному режимі, то для вводу тексту, необхідно перейти в режим вводу. Командою додавання тексту в командному режимі є клавіша [**a**]. При натисканні на цю клавішу відбувається перехід у режим вводу. У даному режимі комп'ютер працює, як друкарська машинка, і користувач може вводити текст у файл. При натисканні клавіші [Enter] відбувається перехід на новий рядок. Працюючи з редактором vi за допомогою клавіш керування курсором, можна переміщатися по тексту, редагуючи різні його частини. Закінчивши набір тексту, можна перейти з режиму вводу в командний режим, натиснувши клавішу [Esc]. Для збереження файлу під час редагування використовується команда **w**, що вводиться в командному рядку vi, яка ініціює запис файлу на диск. Щоб її виконати, потрібно натиснути спочатку клавішу [**:**], включивши тим самим режим редагування командного рядка vi, після чого ввести [**w**] і натиснути клавішу [Enter].

Завершити сеанс редагування можна за допомогою команди **:q**, яка перед виходом ніяких операцій по збереженню файлу не виконує. У цьому відношенні в неї є один серйозний недолік. Якщо з моменту останнього збереження файл коректувався, команда **:q** виконана не буде і вийти з редактора не вдасться. Вийти із ситуації можна, поставивши після команди **:q** специфікатор !. По команді **:q!** відбудеться вихід з редактора vi без збереження змін, внесених у файл у даному сеансі редагування.

Для отримання оперативної підказки призначена команда **:help**, яка вводиться в командному рядку vi. Після слова **help** можна добавити ім’я команди. Клавіша [**F1**] викликає підказку.

Таблиця 1. Команди редактора vi

Команда	**Результат**
Команди переміщення	
h	Переміщення курсору вліво на один символ
l	Переміщення курсору вправо на один символ
k	Переміщення курсору вверх на один рядок
j	Переміщення курсору вниз на один рядок
w	Переміщення курсору вперед на одне слово
b	Переміщення курсору назад на одне слово
e	Переміщення курсору до кінця наступного слова
0	Переміщення курсору в початок рядка
$	Переміщення курсору в кінець рядка
[Enter]	Переміщення курсору на початок наступного рядка
-	Переміщення курсору на початок попереднього рядка

(	Переміщення курсору на початок речення
)	Переміщення курсору в кінець речення; при повторному вводі - переміщення курсору на початок наступного речення
{	Переміщення курсору на початок абзацу
}	Переміщення курсору на наступний абзац
[Ctrl+F]	Прокручування тексту вперед на одну екранну сторінку
[Ctrl+B]	Прокручування тексту назад на одну екранну сторінку
[Ctrl+D]	Прокручування тексту вперед на половину екранної сторінки
[Ctrl+U]	Прокручування тексту назад на половину екранної сторінки
G	Переміщення курсору в останній рядок тексту
***n*G**	Переміщення курсору в n-ий рядок: команда 45G приводить до переміщення курсору в 45-ий рядок
H	Переміщення курсору в перший рядок екрана
M	Переміщення курсору в рядок у середині екрана
L	Переміщення курсору в останній рядок екрана
m*mark*	Установка в тексті мітки *mark*; у якості мітки може застосовуватися будь-яка буква алфавіту
'*mark*	Переміщення в рядок з міткою *mark*
Команди вводу	Після отримання цих команд редактор переходить у режим вводу, вийти з якого можна за допомогою клавіші [Esc]
a	Вставка тексту після курсору
A	Вставка тексту вкінці рядка
i	Вставка тексту перед курсором
I	Вставка тексту на початку рядка
o	Вставка нового порожнього рядка під поточним і перехід у режим вводу в цьому рядку
O	Вставка нового порожнього рядка над поточним і перехід у режим вводу в цьому рядку
Команди видалення	
x	Видаляє символ під курсором
X	Видаляє символ перед курсором
dw	Видаляє слово, на якому знаходиться курсор
dd	Видаляє рядок, на якому знаходиться курсор
D	Видаляє частину рядка, розташованого після курсору
d0	Видаляє текст від курсору до початку рядка
d*comp*	Видаляє зазначений компонент *comp*
J	З'єднує рядок, розташований під курсором, з кінцем поточного рядка
Команди заміни	Всі команди заміни тексту, крім **r**, переводять редактор у режим вводу
s	Видаляє символ і переводить редактор у режим вводу
C	Видаляє залишок рядка, у якому знаходиться курсор, і переводить редактор у режим вводу
cw	Видаляє слово, на якому знаходиться курсор, і переводить редактор у режим вводу
c0	Заміняє текст від курсору до початку рядка
r	Заміняє символ, на якому знаходиться курсор. Після натискання клавіші **r** користувач вводить замінюючий символ. Заміна виконується без переходу в режим вводу, тобто користувач залишається в командному режимі
c*comp*	Заміна зазначеного компонента *comp*
Буфери	**В редакторі vi є 9 нумерованих (1-9) і 26 іменованих буферів (a-z). Для звертання до буфера використовуються подвійні лапки**
"*буква*	Іменований буфер (a, b і т.д. до z)
"*цифра*	Нумерований буфер (1, 2 і т.д. до 9)

Команди переміщення	Переміщення виконується в кілька етапів: спочатку переміщуваний текст потрібно видалити, потім встановити курсор у потрібну позицію, і після цього натисканням клавіші [**p**] вставити текст. (Текст, що видаляється, автоматично поміщається в буфер)
p	Вставка вилученого чи скопійованого тексту в поточну позицію
"*буква*p	Вставка в поточну позицію вмісту іменованого буфера
"*цифра*p	Вставка в поточну позицію вмісту нумерованого буфера
dw p	Видалення слова, а потім переміщення його в позицію, зазначену курсором (клавіша [p] натискається для вставки слова після слова, на якому розміщений курсор)
dcomp	Видалення зазначеного компонента *comp*, а потім переміщення його в місце, зазначене курсором. (Для вставки слова після слова, на якому знаходиться курсор, потрібно натиснути клавішу [**p**])
yy чи **Y**	Копіювання рядка, на якому знаходиться курсор
ycomp	Копіювання зазначеного компонента *comp*
"ayy	Занести поточний рядок в іменований буфер **a**
Команди пошуку	Кожна з двох команд пошуку відкриває в нижньому рядку екрана поле вводу зразка для пошуку. Введення зразка завершується натисканням клавіші [Enter]
/*зразок*	Пошук зразка в прямому напрямку
?*зразок*	Пошук зразка в зворотному напрямку
n	Повторення попередньої операції пошуку в тім же напрямку
N	Повторення попередньої операції пошуку в протилежному напрямку
Команди редагування, що вводяться в командному рядку редактора	
w	Збереження файлу
r *ім'я_файлу*	Вставка тексту з файлу
q	Вихід з редактора
q!	Вихід з редактора без збереження файлу
d	Вихід з редактора з записом, якщо файл був модифікований
Спеціальні символи	
.	Позначає будь-який можливий символ у зразку
*****	Позначає повторювані символи в зразку
[]	Позначає групи символів у зразку
^	Позначає початок рядка
$	Позначає кінець рядка
/<	Позначає початок слова
>/	Позначає кінець слова
Команди підстановки	
s/*зразок/заміна/*	Знаходить у рядку зразок і підставляє замість нього заданий текст
Число-Число **s**/*зразок/заміна/*	Виконує підстановку в зазначеному діапазоні рядків

Засоби та підготовка:
Логін та пароль до робочого місця з компютером, що працює під ОС Linux.

2.Порядок виконання роботи:
6.Запустити систему та залогуватись.
7.Режими роботи редактора та перехід між ними.
а) який режим використовується для вводу тексту?
б) який режим використовується для збереження файлу за завершення роботи vi?
в) який режим є по замовчуванню при старті редактора?
таблиця переходів між редимами:

З режиму	***В режим***	***Команда чи клавіша***
командний	вводу	i (ввід), o (новий рядок), a (додати до текучого рядка)
вводу	командний	Escape
командний	останньої стрічки	:

З режиму	*В режим*	*Команда чи клавіша*
останньої стрічки	командний	Esc або Enter

г) яка клавіша переведе редактор в командний режим з любого іншого?
д) як перейти з режиму вводу в режим останньої стрічки?
5.Створення нових файлів.
Синтаксис команди для створення файлу: vi [options] [filename]
Без задання опцій та назви файлу vi запуститься з новим безіменний файлом.
а) В домашньому каталозі створити катало lab та перейти в нього.
б) відкрити новий файл використовуючи команду vi myfile.
в) перейти в режим вставки тексту використовуючи команду I (або фнший індикатор вибору режиму вводу).
г) написати довільний текст.
д) вийти з режиму вводу натиснувши Esc.
е) натиснути ":" для переходу в режим останньої стрічки
ж) набрати "wq" (w для запису, q для завершення роботи) та натиснути Enter результат виконання даної дії?
з) перевірити командою ls присутність нового створеного файлу Запис та завершення роботи редактора.

Команда	*Значення*
:w	Текучий запис файлу (зберігає зміни та продовжує роботу)
:w new_file	Запис в новий файл та продовження роботи
:wq	Запис та завершення роботи
ZZ	то саме що :wq
:q!	Завершення роботи без збюереження змін
:wq!	Запис та завершення роботи для файлів з атрибутами "тільки для читання

а) в якому режимі вводяться основні команди збереження та завершення роботи?
б) як записати зміни у файлі тільки для читання не завершуючи роботу редактора?
5.Відриття існуючого файлу.
а) відкрити створений файл vi myfile;
б) перемістіть курсор до кінця тексту та натисніть "a" для продовження вводу тексту. Наберіть деякий текст;
в) натисніть Esc для входу в командний режим;
г) перейдіть знову на перший рядок та натисніть "o" для добавлення нового рядка в тексті та введіть ще якийсь текст;
д) запишіть даний файл під новим іменем newfile. Яку команду останньої стрічки використаєте?
е) завершіть роботу редактора без збереженням змін основного файлу. Яку команду останньої стрічки використаєте?
є) перевірте існування нового файлу та порівняйте їх розміри яку команду використаєте.
6.Використання команд редагування.
а) відкрийте свій новий файл. Яку команду використаєте?
б) які команди використаєте для видалення символів, слів та рядків?
г) відмініть деякі останні зроблені дії;
д) скопіюйте перший рядок і вставте його вкінці;
е) запишіть та закрийте файл.
Настройка сесії редактора.
а) використовуючи опції команди set з режиму останньої стрічки поставити параметри показу номерів стрічок та режимів.
8.Використання команд пошуку.
а) перейти на 5 рядок тексту.
б) знайти в тексті стрічку по шаблону проходячи вверх та вниз по тексту.
в) закрити файл без збереження.
9.Витерти створені в лабораторній роботи файлита завершити роботу системи.

3. Порядок оформлення звіту по роботі.

3.1 На титульній сторінці вказати назву університету, кафедри, назву і номер роботи, прізвище, ініціали, номер групи виконавця, прізвище і ініціали викладача, який керував роботою, рік виконання роботи.
3.2 Вказати мету роботи.
3.3 Привести завдання для розробки.

4. Контрольні запитання.

4.1 Який режим є по замовчуванню при старті редактора?
4.2 Як записати зміни у файлі тільки для читання не завершуючи роботу редактора?
4.3 Як перейти з режиму вводу в режим останньої стрічки?
4.4 Як вийти з редактора vi зберігши зміни в записі?

Лабороторна робота №4

Тема: Монтування файлових систем.
Мета: Монтуваття дискет, компакт дисків та розділів жорсткого диску.

1. Підготовчі відомості:
1.1 Команди mount і umount.

Хоча усі файли в системі Linux логічно з'єднані в одне загальне дерево, самі файли розміщаються на різних запам'ятовуючих пристроях, наприклад на жорстких дисках і CD-ROM. Файли, записані на запам'ятовуючих пристроях, організовані у файлові системи. Дерево каталогів в ОС Linux може охоплювати кілька файлових систем, кожна з який розміщена на окремому пристрої. Самі файли організовані в єдине файлове дерево, вершиною якого є кореневий каталог.

Файли тієї чи іншої файлової системи залишаються відділеними від дерева каталогів доти, поки користувач явно не приєднає їх до цього дерева. У кожній файловій системі файли організовані в окреме дерево каталогів. Це дерево можна розглядати як піддерево, яке необхідно приєднати до основного дерева каталогів. Наприклад, на дискеті з файлами ОС Linux - своє дерево каталогів. Це піддерево потрібно приєднати до основного дерева, розташованому в розділі жорсткого диска. Поки це не зроблено, доступу до файлів на дискеті не буде.

Приєднання файлової системи, розташованої на запам'ятовуючому пристрої, до основного дерева каталогів називається **монтуванням** пристрою. Монтування пристрою здійснюється командою **mount**. Для того щоб одержати можливість працювати з файлами, записаними на CD-ROM, цей пристрій потрібно спочатку змонтувати. Операція монтування припускає приєднання дерева каталогів, що знаходиться на пристрої пам'яті, до зазначеного користувачем каталогу. Лише після цього можна перейти в приєднаний каталог і звертатися до його файлів. Монтувати файлові системи може тільки привілейований користувач **root**. Ця задача відноситься до функцій системного адміністратора, і звичайний користувач її виконувати не може. Для того щоб монтувати файлову систему, потрібно ввійти в систему як привілейований користувач.

Команда mount приймає два аргументи: ім'я пристрою, через яке Linux одержує доступ до файлової системи, і каталог у файловій структурі, до якого приєднується нова файлова система. Місце монтування - це каталог, до якого користувач хоче приєднати файли, що знаходяться на запам'ятовуючому пристрої. Пристрій - це спеціальний файл пристрою, за допомогою якого система одержує доступ до апаратних пристроїв. Команда **mount** має наступний синтаксис:

 # mount пристрій місце_монтування

Файли пристроїв знаходяться в каталогах **/dev** і зазвичай мають скорочені імена, що закінчуються номером пристрою. Наприклад, **fd0** може позначати перший дисковод гнучких дисків, приєднаний до системи. У Linux-системах, що працюють на PC, розділи жорсткого диска мають префікс hd, за яким йде буквений символ, що позначає цей диск, і номер розділу. Наприклад, hda2 позначає другий розділ першого жорсткого диска.

Для того щоб до файлової системи був можливий доступ, вона повинна бути змонтована. Навіть файлову систему, розташовану в розділі жорсткого диска, потрібно монтувати командою **mount**. При інсталяції Linux і створенні на жорсткому диску розділу Linux система автоматично конфігурується на монтування основних файлових систем при кожнім запуску. Для дискет і компакт-дисків така можливість не передбачена, і їх прийдеться монтувати явно.

 # mount /dev/fd0 /mydir

Перед тим як зупиняти систему, необхідно демонтувати всі змонтовані файлові системи. Основні файлові системи демонтуються автоматично. Якщо користувач хоче замінити змонтовану файлову систему іншою, спочатку потрібно демонтувати першу явно. Нехай, користувач змонтував файли, що містилися на дискеті, а тепер хоче замінити її на іншу. Для цього потрібно спочатку демонтувати файлову систему встановленої дискети. Файлова система демонтується командою **umount**. Як аргументи ця команда використовує ім'я пристрою і каталог, у якому воно було змонтовано. От синтаксис команди **umount**:

 # umount пристрій місце_монтування

У наступному прикладі демонтується гнучкий диск, змонтований у каталозі /mydir:

 # umount /dev/fd0

Для команди **umount** встановлене одне істотне обмеження. Не можна демонтувати файлову систему, у якій користувач працює в даний момент. Якщо перейти в який-небудь каталог файлової системи і потім спробувати демонтувати її, то з'явиться повідомлення про помилку - повідомлять про те, що файлова система

зайнята. Спочатку потрібно вийти з файлової системи, яку потрібно демонтувати і тільки після цього можна виконати команду **umount**.

 # mount /dev/hdc /mnt/cdrom

 # umount /mnt/cdrom

umount: /dev/hde: device is busy

 # cd /root

 # umount /mnt/cdrom

Файлові системи на всіх запам'ятовуючих пристроях займають весь виділений для них обсяг. Визначити, скільки вільного простору є у файловій системі, можна за допомогою команди **df**. Вона видає список усіх файлових систем по іменах пристроїв, повідомляє їх розмір і місця монтування.

 # df

Filesystem	1024-blocks	Used	Available	Capacity	Mounted on
/dev/hda3	297635	169499	112764	60%	/
/dev/hdal	205380	182320	23060	89%	/mnt/dos
/dev/hdc	637986	637986	0	100%	/mnt/cdrom

Команду **df** можна використовувати і для одержання інформації про те, до якої файлової системи відноситься той чи інший каталог. Для цього команду **df** потрібно ввести з ім'ям каталогу (для поточного каталогу - df .).

 # df.

Filesystem	1024-blocks	Used	Available	Capacity	Mounted on
/dev/hda3	297635	169499	112764	60%	/

1.2 Монтування компакт-дисків

У ОС Linux каталог **/mnt/cdrom** зарезервований для файлових систем на компакт-дисках. Можна побачити рядок для цього каталогу у файлі **/etc/fstab**. Щоб змонтувати компакт-диск, потрібно лише ввести команду **mount** і ім'я каталогу **/mnt/cdrom**. Ім'я пристрою вказувати не треба. Після монтування можна звертатися до компакт-диску через каталог /mnt/cdrom.

 # mount /mnt/cdrom

Для того щоб поміняти диски, спочатку потрібно демонтувати компакт-диск, встановлений у дисководі. Потім потрібно вставити новий диск і явно змонтувати його.

 # umount /mnt/cdrom

Тепер можна вийняти компакт-диск, уставити новий і дати команду mount:

 # mount /mnt/cdrom

Щоб змонтувати компакт-диск в іншому каталозі, необхідно вказати в команді **mount** ім'я пристрою. Нижче приведений приклад, у якому компакт-диск, що знаходиться в дисководі, монтується в каталозі **/mydir**. У цьому прикладі для компакт-диску файл пристрою має ім'я **/dev/hdc**.

 # mount /dev/hdc /mydir

Ім'я файлу пристрою компакт-диску залежить від типу дисковода. Якщо це IDE, то ім'я пристрою буде мати той же префікс, що і розділ жорсткого диска IDE, hd. Воно позначається символом, що відрізняє його від інших IDE-пристроїв. Наприклад, IDE-дисковод компакт-дисків, підключений до допоміжного IDE-порту, може мати ім'я hdc. IDE-дисковод, підключений як підлеглий (slave) до допоміжного порту, може мати ім'я hdd. Фактичне ім'я буде визначено при інсталяції дисковода, як це було при інсталяції Linux-системи. Імена SCSI-дисководів компакт-дисків складаються з інших компонентів. Вони починаються з букв sd (скорочення від SCSI drive), за якими йде спеціальний символ. Такий дисковод може мати, наприклад, ім'я sdb чи sda. Ім'я дисковода визначається при інсталяції системи. Його можна довідатися з файлу **/etc/fstab**.

1.3 Монтування і форматування дискет

Дисковод гнучких дисків позначається - /dev/fd0. Якщо дисководів гнучких дисків декілька, вони будуть позначатися як fdl, fd2 і т д. Монтувати пристрій можна в будь-який каталог, але у версії Red Hat уже

створений зручний каталог для файлів дискет – **/mnt/floppy**. У наступному прикладі файли дискети монтуються в цей каталог.

 # mount /dev/fd0 /mnt/floppy

Потрібно пам'ятати про те, що монтується не дисковод, а конкретна дискета. Просто вийняти її і вставити іншу не можна. Команда **mount** приєднала файли, що знаходяться на дискеті, до основного дерева каталогів. Для того щоб поміняти дискету, спочатку потрібно демонтувати дискету, вставлену в дисковод. Потім потрібно вставити нову дискету і явно змонтувати її файли.

Для демонтування використовується команда **umount**. В операції **umount** можна вказати або каталог, у якому змонтований гнучкий диск, або пристрій /dev/fd0.

 # umount /dev/fd0

 # umount /mnt/floppy

При зупинці системи здійснюється автоматичне демонтування всіх змонтованих дисків. Явно демонтовувати їх не потрібно.

Для форматування дискети служить команда **mkfs**. Як аргументи вона використовує ім'я пристрою і число блоків пам'яті на дискеті. При кількості блоків 1400 дискета форматується на 1,44 Мбайт. Для задання файлової системи використовується опція **–t** *файлова_система.* У наступному прикладі дискета форматується на 1,44 Мбайт:

 # mkfs -t ext3 /dev/fd0 1400

1.4 Монтування розділів жорсткого диска

Розділи жорсткого диску для Linux і MS-DOS можна монтувати за допомогою команди **mount**. Зручніше, однак, монтувати їх автоматично за допомогою файлу **/etc/fstab.** Ті розділи жорсткого диска для Linux, які створились під час інсталяції, вже автоматично змонтовані. Для монтування розділу жорсткого диску використовується команда **mount** з ім'ям розділу і каталог, у який потрібно змонтувати даний розділ. Ім'я розділу складається з префікса (hd для IDE-дисків і sd для SCSI-дисків), мітки диска і номера розділу. Наприклад, hda2 позначає другий розділ першого IDE-диску, а sdbs - третій розділ другого SCSI-диску. У наступному прикладі розділ жорсткого диску для Linux монтується на пристрої **/dev/hda4** у каталозі **/mnt**.

 # mount -t ext3 /dev/hda4 /mnt

Для монтування розділу MS-DOS використовується команда **mount** з опцію **-t msdos,** яка задає тип файлової системи - MS-DOS. У наступному прикладі користувач монтує розділ жорсткого диску MS-DOS, **/dev/hda1**, у каталог **/mnt/dos**. Каталог **/mnt/dos** - стандартне місце монтування для файлових систем MS-DOS, однак, їх можна монтувати в будь-якому іншому каталозі. Перед виконанням команди **mount** потрібно перевірити чи створений каталог у який монтується розділ жорсткого диску .

 # mount -t msdos /dev/hda1 /mnt/dos

Якщо потрібно включити у файлову структуру новий розділ, спочатку необхідно створити цей розділ за допомогою команди **fdisk** чи команди **cfdisk**, а потім відформатувати його за допомогою команди **mkfs**. Після цього, цей розділ можна монтувати.

1.5 Автоматичне монтування файлових систем: файл fstab

Для того щоб ОС Linux автоматично монтувала файлову систему, що існує в новому розділі жорсткого диска, потрібно додати її ім'я у файл **fstab**. Файл **fstab** розташований у каталозі **/etc**. У ньому перераховані файлові системи, що монтуються командою **mount** з опцією **-a**. Ця команда знаходиться у файлі **/etc/rc.d/rc.boot**. Команди даного файлу виконують операції по ініціалізації системи. Вони виконуються при кожнім завантаженні системи. При зупинці системи виконується команда **umount -a**, що демонтує усі файлові системи, перераховані у файлі **/etc/fstab**. Команда **umount -a** знаходиться у файлі **/etc/rc.d/init/halt**, що містить команди, які виконуються при кожній зупинці системи. Таким чином, усі файлові системи, що вказуються у файлі **/etc/fstab**, автоматично монтуються при запуску системи і демонтуються при її закритті.

Елемент файлу **fstab** містить кілька полів, розділених пробілами чи знаками табуляції. Перше поле - ім'я монтованої файлової системи. Воно зазвичай починається з **/dev**, наприклад **/dev/hda3**, - третій розділ жорсткого диска. Наступне поле - каталог у файловій структурі, до якого потрібно приєднати файлову систему, що знаходиться на даному пристрої. Третє поле - тип монтованої файлової системи. Тип розділу жорсткого диску для стандартної файлової системи Linux – ext3. У наступному прикладі показаний рядок файлу **fstab**, що відповідає основному розділу жорсткого диска для Linux. Він монтується в кореневому каталозі (/) і має тип ext3.

/dev/hda3 / ext3 defaults 0 1

У полі, що стоїть після типу файлової системи, вказуються різні опції монтування. Є стандартний набір опцій, встановлюваних по замовчуванню; усі їх можна задати введенням однієї опції **defaults**. Інші опції в списку розділяються комами (без пробілів). Опція defaults позначає пристрій читання/запису, асинхронне, блок-орієнтоване, без можливості монтування для звичайних користувачів, з можливістю виконання на ньому програм. Для CD-ROM указуються всього дві опції, **ro** і **noauto**. Опція **ro** показує, що файли, що знаходяться на цьому пристрої, призначені тільки для читання, **noauto** - що воно автоматично не монтується. Опція **noauto** використовується як для CD-ROM, так і для дискет, щоб вони не монтувалися автоматично, оскільки не відомо, чи будуть вони встановлені при запуску. Нижче приводиться приклад записів для CD-ROM і дискет. Тип файлової системи на CD-ROM iso9660.

/dev/fd0 /mnt/floppy ext3 defaults,noauto 0 0

/dev/hdc /mnt/cdrom iso9660 ro,noauto 0 0

Останні два поля містять цілі значення. Перше використовується командою **dump** для визначення періодичності резервного копіювання файлової системи. Останнє використовується командою **fsck** для визначення необхідності перевірки системи і порядку можливої перевірки. Якщо значення поля - 1, то це кореневий розділ. Значення 0 говорить про те, що перевіряти файлову систему при завантаженні не потрібно.

Нижче приведена копія файлу **/etc/fstab**. Перший його рядок - коментар. Файлова система **/proc** - це спеціальна файлова система, яку ОС Linux використовує для керування системними процесами. Ніякому реальному пристрою вона не відповідає.

# device	mountpoint	filesystemtype	options	dumpm	fsckorder
/dev/hda3	/	ext3	defaults	0	1
/dev/hdc	/mnt/cdrom	iso9660	ro,noauto	0	0
/dev/fd0	/mnt/floppy	ext2	defaults,noauto	0	0
/proc	/proc	proc	defaults		
/dev/hda2	/none	swap	sw		
/dev/hda1	/mnt/dos	msdos	defaults	0	0
/dev/hda4	/mnt/win	vfat	iocharset=koi8-r,codepage=866,rw,noexec	0	0
/dev/hdb1	/mnt/zip	vfat	rw,umask=111,gid=100,iocharser=koi8,noauto	0	0

Щоб можна було читати і створювати файли з російськими іменами рекомендовано для файлових систем **msdos** і **vfat** проводити монтування з наступними параметрами:

/dev/fd0 /mnt/a: msdos rw,umask=111,gid=100,iocharser=koi8,noauto 0 1

/dev/hda1 /mnt/win vfat rw,umask=111,gid=100,iocharser=koi8,noauto 0 1

Щоб вказати у файлі **/etc/fstab** нову файлову систему, можна або відредагувати його вручну, або скористатися утилітою **fstool**, що попросить увести відповідну інформацію.

Щоб система автоматично монтувала розділи MS-DOS при запуску Linux потрібно ввести у файл **/etc/fstab** запис для кожного монтованого розділу MS-DOS.

/dev/hda1 /mnt/dos msdos defaults 0 0

Розділ, для якого у файлі **/etc/fstab** є запис, можна монтувати тільки в каталозі, зазначеному в цьому записі. Ім'я файлу пристрою вводити не потрібно. Програма **mount** знайде запис, що відповідає цьому розділу, у файлі **fstab** (по імені каталогу) і в такий спосіб визначить ім'я пристрою. Наприклад, щоб демонтувати DOS-розділ **/dev/hda1** у попередньому прикладі, команді u**mount** потрібно вказати лише каталог, у якому він змонтований. У даному випадку це /mnt/dos.

 # umount /mnt/dos

Якщо файл **/etc/fstab** зіпсований, то система завантажиться в режимі супроводу і надасть доступ до розділів тільки для читання. Щоб одержати доступ на читання і запис і виправити файл **fstab**, необхідно перемонтувати основний розділ. Цю операцію виконує наступна команда:

 # mount -n -0 remount,ro /

2. Порядок виконання роботи:

Запустити систему та залогуватись.
Підготовка до монтування.
а) переглянути каталог точки монтування файлових систем /mnt на предмет наявності каталогів floppy та cdrom.
б) при їх відсутності створити вказані каталоги а також створити каталог hd_t.
1. Монтування дискет.
а) вставити дискету в дисковод та примонтувати командою mount -t msdos /dev/fd0 /mnt/floppy
б) перевірити примонтованість командою df. Короткий опис виводу на термінал.
в) проглянути вміст дискети.
г) демонтувати дискету.
2. Монтування компакт-дисків.
а) який тип файлової системи використовується на компакт дисках?
б) проглянути на який пристрій системи підключений дисковод компакт дисків dmesg|grep hd. Короткий опис виводу на термінал з висновком.
в) вставити компакт диск в дисковод та змонтувати.
г) перевірити змонтованість командою df –T. Короткий опис виводу на термінал.
д) попробувати натиснути кнопку eject на дисководі компакт дисків. Описати результат даної дії.
е) демонтувати компакт диск.
є) змонтувати компакт диск використовуючи параметри файлу автоматичного монтування /etc/fstab: mount /mnt/cdrom.
ж) перевірити змонтованість командою df –T. Короткий опис виводу на термінал.
з) демонтувати диск командою eject.
3. Монтування розділів жорсткого диску.
а) змонтувати вказаний інструктором розділ жорсткого диску в каталог /mnt/hd_t.
б) перевіритизмонтованість командою df –T. Короткий опис виводу на термінал.
в) демонтувати даний розділ.
Завершити роботу системи.

3. Порядок оформлення звіту по роботі.

3.1 На титульній сторінці вказати назву університету, кафедри, назву і номер роботи, прізвище, ініціали, номер групи виконавця, прізвище і ініціали викладача, який керував роботою, рік виконання роботи.
3.2 Вказати мету роботи.
3.3 Привести завдання для розробки.

4. Контрольні запитання.

4.1 Якою командою монтують файлову систему?
4.2 Як подивитись вміст дискети?
4.3 Як розмонтувати файлову ситему?
4.4 Якою командою можна перевірити примонтованість файлової системи?

Лабороторна робота №5

Тема: Робота з архіваторами, процеси і роботи.
Мета: Проводити резервне архівування даних та користуватись основними архіваторами, керування процесами і роботами.

1. Підготовчі відомості:

1.1 Програма tar

Утиліта tar призначена для резервного копіювання даних та створення архівів файлів і каталогів. За допомогою цієї програми можна архівувати файли, обновляти їх в архіві і вводити в цей архів нові файли. Можна архівувати і цілі каталоги з усіма їхніми файлами і підкаталогами. При необхідності всі ці файли і підкаталоги можна відновити з архіву. Програма tar призначалася для створення архівів на магнітних стрічках, звідси і назва tar (tape archive, тобто "архів на стрічці"). Архів можна створювати на будь-якому пристрої, наприклад на дискеті чи в архівному файлі на диску.

Для архівування файлів на визначеному пристрої чи у визначеному файлі служить опція **f** з ім'ям пристрою чи файлу. Синтаксис команди **tar** з опцією **f** наведений нижче

```
$ tar опції f ім'я_архіву.tar імена_файлів_і_каталогів
```

Ім'я пристрою чи файлу часто вибирають для імені архіву. При створенні файлу для tar-архіву до імені цього файлу зазвичай додається розширення .tar. Це умовна позначка; вона не обов'язкова. У команді можна вказати скільки потрібно імен файлів. Якщо зазначене ім'я каталогу, то в архів включаються і всі підкаталоги цього каталогу.

В табл. 1 перераховані основні опції команди tar.

Таблиця 1. Опції команди tar.

Команда чи опція	Призначення
tar *опції файли*	Створює резервні копії файлів в архівному файлі, на стрічці чи іншому пристрої
tar *опції* f *ім'я_архіву список_файлів*	Створює резервні копії файлів у файлі *ім'я_архіву; список файлів* може містити імена файлів і каталогів
c	Створює новий архів
t	Видає список файлів, наявних в архіві
r	Додає файли в архів
u	Обновляє архів новими і виправленими файлами. Додає лише ті файли, що змінилися після останнього архівування, і файли, що в архіві відсутні
w	Очікує від користувача підтвердження на архівування кожного файлу; дозволяє обновляти архів вибірково
x	Розархівування архіву
m	Витягає файли з архіву, не змінюючи дату його останньої модифікації
M	Створює багатотомний архів, що може зберігатися на декількох носіях
v	Відображає кожне ім'я файлу по мірі архівування
z	Здійснює стиск і розпакування архівованих за допомогою програми **gzip** файлів

Для створення архіву служить опція **c**. У сполученні з опцією **f** опція **c** призводить до створення архіву в файлі чи на пристрої. Ця опція ставиться безпосередньо перед опцією **f**. У наступному прикладі каталог **mydir** і всі його підкаталоги зберігаються у файлі **myarch.tar**.

```
$ tar cf myarch.tar mydir
```

Для розархівування архіву використовується команда **tar** з опцією **x**. Опція **xf** дозволяє витягати файли чи каталоги з конкретного архіву. При розархівуванні формуються всі підкаталоги. У наступному прикладі розархівовуються усі файли і підкаталоги з архіву **myarch.tar**.

```
$ tar xf myarch.tar
```

Для додавання файлів в існуючий архів служить опція **r**. У приведеному нижче прикладі користувач додає файли з каталогу **letters** в архів **myarch.tar**.

```
$ tar rf myarch.tar letters
```

Для поновлення архіву новими файлами чи новими версіями існуючих файлів використовується опція **u**. Програма **tar** порівнює час останньої зміни кожного заархівованого файлу і відповідного файлу в каталозі і копіює в архів усі файли з більш пізньою датою модифікації. У наступному прикладі користувач обновляє архів **myarch.tar**, вводячи в нього всі змінені і наново створені в каталозі **mydir** файли.

```
$ tar uf myarch.tar mydir
```

Для перегляду архіву використовується команда **tar** з опцією **t**. У наступному прикладі показано, як за допомогою цієї команди можна викликати список усіх файлів, що зберігаються в архіві **myarch.tar**.

```
$ tar tf myarch.tar
```

Для створення резервних копій файлів на певному пристрою потрібно вказати ім'я цього пристрою як ім'я архіву. У наступному прикладі користувач створює архів на дискеті в пристрої **/dev/fd0** і копіює у нього усі файли з каталогу **mydir**.

```
$ tar cf /dev/td mydir
```

Для розархівування архіву потрібно використати команду **tar** з опцію **xf**.

```
$ tar xf /dtv/fd0
```

Якщо архівовані файли займають більше місця, ніж є на носії, наприклад на дискеті, можна створити багатотомний tar-архів (опція **M**). При архівуванні файлів на дискеті з використанням опції **M** у випадку заповнення дискети програма **tar** запропонує користувачу вставити нову дискету.

```
$ tar cMf /dev/fd0 mydir
```

Щоб розпакувати архів, записаний на декількох дискетах, потрібно вставити першу дискету в дисковод і ввести команду **tar** з опціями **x** і **M**. Програма вкаже, коли треба вставити наступну дискету.

```
$ tar xMf /dev/fd0
```

При використанні команди **tar** операція стиску архівних файлів не виконується. Якщо потрібно стиснути файли, необхідно використати команду **tar** з опцією **z**, щоб дати вказівку викликати утиліту gzip. У цьому випадку спочатку програма **gzip** виконує стиск, а потім **tar** архівує файли. Та ж опція z забезпечить виклик gzip для розпакування файлів при витягу їх з архіву.

```
$ tar czf myarch.tar mydir
```

У багатьох випадках архів створюється, щоб переслати по мережі кілька файлів у виді одного tar-файлу. Для скорочення часу передачі розмір цього архіву повинен бути по можливості невеликим. Щоб домогтися цього, можна за допомогою утиліти **gzip** стиснути архівний tar-файл, зменшивши його розмір, а потім переслати стиснуту версію. Одержувач розпакує його і відновить файл. У результаті застосування утиліти **gzip** до tar-файлів часто виходять файли з розширенням .tar.gz. Розширення .gz додається до стиснутого gzip-файлу.

Якщо користувач визначив пристрій, який буде використовуватись по замовчуванню (наприклад стрічку) і хоче створити на ньому архів, то команда **tar** спрощується: в ній не потрібно задавати ні опцію **f** ні ім’я пристрою чи файлу. Це зручно при створенні резервних копій файлів. Ім'я пристрою по замовчуванню зберігається у файлі **/etc/default/tar**. Синтаксис команди **tar**, що припускає використання пристрою, заданого по замовчуванню

```
$ tar опція імена_каталогів_і_файлів
```

Якщо зазначене ім'я каталогу, то в архів включаються всі його підкаталоги.

У представленому нижче прикладі каталог **mydir** із усіма підкаталогами зберігається на носію по замовчуванню

```
$ tar c mydir
```

1.2 Програма gzip

За допомогою утиліти **gzip** здійснюються стиск файлів. При стиску як аргумент вводиться ім'я файлу. Цей файл заміняється стиснутою версією з розширенням .gz.

```
$ gzip mydata
```

Для розпакування gzip-архіву потрібно ввести або команду **gzip** з опцією **-d**, або команду **gunzip**. Ці команди призводять до розпакування файлу з розширенням .gz і заміні його розпакованою версією з тим же ім'ям, але без розширення .gz. При використанні команди **gunzip** не потрібно навіть уводити розширення .gz.

```
$ gunzip mydata.gz
```

Перелік основних опцій команди **gzip** приведений в табл. 2.

Таблиця 2. Опції команди **gzip**

Опція	**Призначення**
-c	Посилає стиснуту версію файлу на стандартний вивід; кожен зазначений файл стискується окремо `$ gzip -c mydata preface > myffiles.gz`

-d	Розпаковує стиснутий файл; можна також використовувати команду gunzip
	`$ gzip -d myffiles.gz`
	`$ gunzip myffiles.gz`

-h	Видає перелік довідкової інформації
-l *список_файлів*	Видає розмір кожного з зазначених файлів (у стиснутому і не стиснутому виді) `$ gzip -l myffiles.gz`
-r *ім'я_каталогу*	Стискає усі файли в зазначених каталогах. При використанні цієї опції з програмою gunzip стиснуті файли з зазначеного каталогу будуть розпаковані.
-v *список_файлів*	Повідомляє степінь стискання по кожному обробленому файлі
-число	Визначає швидкість і ступінь стиску. Діапазон чисел - від -1 до -9. Менше число означає більш високу швидкість, але менший ступінь стиску - у підсумку виходить великий файл, що швидко стискується і розпаковується. Число -1 означає найшвидший стиск, але і максимальний розмір. Число -9 дає дуже маленький файл, що стискується і розпаковується повільно. По замовчуванню приймається -6.

Можна стискати й архівірувані файли. Ця операція дає в результаті файли з расширением .tar.gz. Стиснуті архівовані файли часто використовуються для передачі дуже великих файлів по мережах.

```
$ gzip myarch.tar
```

Файли, що входять в архів, можна стискати і по одинці, використовуючи команду tar з опцією **z**, що викликає утиліту gzip. У цьому випадку файл спочатку стискається, а потім поміщується в архів. Слід зазначити, однак, що архіви з файлами, стиснутими з застосуванням опції **z**, відновленню не підлягають, і додавати в них файли не можна. Усі файли необхідно стискати одночасно і додавати теж одночасно.

Для створення стиснутих файлів можна також скористатися командами **compress** і **uncompress**. В утиліті **compress** використовується інший формат стиску. У результаті її використання утворяться файли з розширенням .Z. Команди **compress** і **uncompress** застосовуються не дуже широко, але файли з розширенням .Z іноді зустрічаються. Для розпакування файлу з розширенням .Z можна використовувати не лише команду **uncompress**, але і команду **gunzip**. Однак **gzip** є стандартною утилітою стиску, тому замість команди **compress** по можливості варто використовувати саме **gzip**.

1.3 Керування процесами

Кожна задача, яка виконується в системі, розглядається ОС Linux як **процес**, якому привласнюються номер і ім'я. Система Linux може виконувати кілька процесів одночасно так само, як і працювати одночасно з декількома користувачами. До процесів відносяться не тільки команди, які виконуються користувачем, але і всі завдання, які система повинна виконувати для забезпечення власної працездатності.

Команди, що надходять від користувачів, часто називають **завданнями**, щоб відрізнити їх від системних процесів. Коли користувач вводить команду, вона стає завданням, яке повинна виконати система. У командному інтерпретаторі використовується ряд спеціальних операцій керування процесами, що дозволяють користувачу контролювати виконання своїх завдань.

Щоб побачити список процесів поточного користувача використовується команда **ps**.

```
$ ps
PID   TTY   TIME   COMMAND
523   tty24 0:05   ah
567   tty24 0:01   lpr
570   tty24  0:00  ps
```

У даному прикладі створюється список процесів, запущених користувачем. У стовпці **PID** зазначений системний номер процесу (ідентифікатор процесу). Стовпець **TTY** відображає ідентифікатор термінала. У

стовпці **TIME** відображається час, необхідний для виконання процесу. А в стовпці **COMMAND** зазначене ім'я процесу.

Щоб побачити всі виконувані системою процеси потрібно виконати команду **ps** з опцією **–x**, а для перегляд всіх виконуваних процесів системи з виводом додаткової інвормації потрібно використати опцію **–xl**.

Для візуального перегляду дерева процесів використовується команда **pstree**. Для ітерактивного контролю за процесами використовується програма **top**.

Запуск процесів можна проводити як в терміналі так і в фоновому режимі. Таким чином процес буде виконуватись, але на термінал не буде надходити люба інформація з цього процесу. Це робиться з допомогою останнього символа & до командної стрічки запуску процесу.

Процеси можна запускати у фоновому режимі, одночасно виконуючи інші команди. Таким чином процес буде виконуватись, але на термінал не буде надходити інформація з цього процесу. Для того щоб виконати команду у фоновому режимі, необхідно наприкінці її поставити амперсанд (&). При цьому система видає на екран номер завдання користувача і системний номер процесу. Номер завдання, поміщений у квадратні дужки – це номер, по якому користувач може вказувати конкретне завдання. Номер процесу — це номер, під яким завдання проходить у системі. У наступному прикладі у фоновий режим переводиться команда друку файлу **mydata**.

```
$ lpr mydata &
[1] 534
```

У фоновий режим можна переводити кілька команд. Кожна з них розглядається як завдання й одержує ім'я і номер завдання. Для одержання списку завдань, виконуваних у фоновому режимі, застосовується команда **jobs**. Кожна позиція списку складається з номера завдання (у квадратних дужках), вказівки на те, зупинене воно чи виконується, і імені завдання. Знак + позначає завдання, яке виконується в даний момент, а знак – позначає наступне завдання, яке буде виконуватись.

```
$ jobs
[1] + Running lpr mydata
[2] - Running  cat *.c  > myprogs
```

Після виклику на виконання будь-якої команди система Linux повідомляє, які фонові завдання — якщо такі є — завершені на даний момент. При цьому система не перериває ніяких операцій, для повідомлення про завершення завдання. Щоб негайно одержати повідомлення про завершення певного завдання потрібно ввести команду **notify**. Як параметр у цій команді використовується номер завдання, причому перед ним ставиться символ %. Коли це завдання завершується, система переривається поточну операцію і повідомляє користувача про те, що завдання виконане. У наступному прикладі системі дається вказівка повідомити користувача про закінчення виконання завдання 1.

```
$ notify %1
```

При завершенні виконання фонового процесу видається відповідний звіт:

```
[1]+ Done lpr mydata
```

Зупинити завдання, яке виконується у фоновому режимі можна за допомогою команди **kill**. Як параметр у команді **kill** використовується номер користувальницького завдання чи номер процесу. Перед номером користувальницького завдання повинен бути знак %. Довідатися номер завдання можна за допомогою команди **jobs**, а номер процесу – за допомогою команди **ps**.

```
$ kill %2
$ kill 567
```

Виконання завдання можна перервати командою [Ctrl+Z]. З її допомогою виконання завдання відкладається до його перезапуску. Завдання не завершується, а просто залишається припиненим. У потрібний момент його виконання можна продовжити в пріоритетному чи у фоновому режимі, для чого використовуються команди **fg** чи **bg**.

Для переводу завдання з фонового режиму в пріоритетний використовується команда **fg** (скорочення від foreground). Якщо у фоновому режимі виконується тільки одне завдання, його можна перевести в пріоритетний режим, указавши команду **fg** без параметрів. Якщо ж у фоновому режимі виконується кілька завдань, необхідно при використанні команди додавати номер завдання. Цей номер вказується після команди **fg**.

```
$ fg %2
```

```
cat *.c  > myprogs
```

Команда **bg** переводить перерване завдання у фоновий режим. Іноді потрібно перевести завдання, яке виконується в пріоритетному режимі, у фоновий режим. Однак зробити це в один прийом не можна. Спочатку потрібно зупинити виконання завдання командою [Ctrl+Z], а потім перевести його у фоновий режим командою **bg**. У наступному прикладі поточна команда (видача списку і перенапрямок файлів з розширенням с) спочатку переривається командою [Ctrl+Z]. Потім це завдання переводиться у фоновий режим.

```
$ cat *.c > myprogs
^z
$ bg
```

Засоби та підготовка:
Логін та пароль до робочого місця з компютером, що працює під ОС Linux.

2. Порядок виконання роботи:
Запустити систему та залогуватись.
2.1Архіватори.
Робота з tar.
Tar – системна утиліта резервного копіювання даних на магнітні носії та з можливістю створення архівних файлів у файлових системах.
Синтаксис команди: tar [option] [file]
Основні ключі:
-A – додати файли до архіву;
-c – створити новий архів;
-u – обновити архів;
-x – розархівувати файли;
-t – проглянути вміст архіву.
а) створити в домашньому каталозі підкаталог arj та в ньому файли arj.doc, arj.lab з деяким текстом.
б) в каталозі arj заархівувати всі файли в архів tar.arj і розмістити його в даному каталозі.
в) переглянути і зробити аналіз розмірів файлів всумі та архіву. Короткий опис команд та висновки.
г) заархівувати весь каталог з вмістом arj в архіви tar.arj та розмістити його в домашньому каталозі.
д) витерти створені архіви.
4.Робота з gzip та bzip2.
Синтаксис команди: gzip{bzip2} [options] [-S suffix] file
а) створити архіви з файлів в каталозі arj з суфіксом gzip архіватором gzip.
б) переглянути вміст каталогу на предмет архівування кожного з них з відповідним суфіксом.
в) розархівувати файли використовуючи gzip.
г) повторитиархівування архіватором bzip2.
д) розархівувати файли використовуючи bzip2.
6.Робота з zip.
Синтаксис команди: zip [options] arc_file file_list
Основні ключі:
-u – обновити архів;
-r – рекурсія в підкаталоги;
-e – шифрування архіву з паролем;
-x – розархівування файлів.
а) в каталозі arj заархівувати всі файли в архів arj.zip і розмістити його в даному каталозі.
б) переглянути і зробити аналіз розмірів файлів всумі та архіву. Короткий опис команд та висновки.
в) заархівувати весь каталог з вмістом arj в архіви arj.zip та розмістити його в домашньому каталозі.
г) видалити створені архіви та каталог arj.

2.2 Процеси і роботи.
а) переглянути які процеси запущені в даний момент.
б) переглянути всі процеси.
в) Запустити ‘y’ на стандартний вивід. Команда yes.
г) Зупинити процес за допомогою комбінації клавіш ctrl-C.
д) перемістити процес в фоновий режим : yes>/dev/null &
е) перевірити стан виконуваного процесу, команда jobs. Пояснити результати виводу команди?

є) завершити роботу по номеру виконуваної роботи.
з) перевірити чи завершили роботу.
ж) запустити роботу і зувершити її по номеру ідентифікатора процесу (PID).
і) запустити роботу і помістити її на задній план.
й) перевірити чи робота виконується на задньому плані.
к) якщо робота виконується у фоновому режимі то перемістити її на передній план.
л) запустити дві роботи на виконання і зупинити першу по номеру роботи, а другу по номеру ідентифікатора процессу.
Завершити роботу системи

3. Порядок оформлення звіту по роботі.

3.1 На титульній сторінці вказати назву університету, кафедри, назву і номер роботи, прізвище, ініціали, номер групи виконавця, прізвище і ініціали викладача, який керував роботою, рік виконання роботи.
3.2 Вказати мету роботи.
3.3 Привести завдання для розробки.

4. Контрольні запитання.

4.1Яка команда для архівування ?
4.2 Синтаксис команди архівування?
4.3 Які команди розархівування даних?
4.4 Які основні ключі архівування?
4.5 Яка командою перевіряють які процеси запущені в системі?
4.6 Якою командою роботу переводять у фоновий режим?
4.7 Якою командою роботу переводять на передній план?
4.8 Якою командою зупиняють роботу?

Лабороторна робота №6

Тема: Написання сценаріїв shell.
Мета: Навчитися створювати shell сценарії.

1. Підготовчі відомості:

1.1 Основні поняття

Об'єднавши декілька команд Linux у shell-сценарій, користувач може істотно заощадити час при виконанні важливих і складних завдань. Інтерпретатор shell пропонує користувачу безліч інструментів створення shell-сценаріїв. У цих сценаріях використовуються змінні, значення яких можна задавати безпосередньо в сценарії чи вводити інтерактивно, умовні оператори, що змінюють хід виконання сценарію, а також цикли, що забезпечують багаторазове виконання групи команд Linux. У сценарії можна застосовувати арифметичні і логічні вирази — точно так само, як це робиться в інших мовах програмування.

1.2 Змінні в сценарії

У shell-сценарії можна використовувати змінні так, як і в будь-якій програмі, написаній на іншій мові програмування, наприклад C. Змінна визначається в командному інтерпретаторі shell при першому використанні її імені. Ім'я змінної може складатися з будь-якої кількості буквених символів, включаючи символ підкреслення. Ім'я може містити і цифри, однак не повинно починатися з цифри. Значення присвоюється змінній за допомогою оператора присвоювання. Спочатку вводиться ім'я змінної, потім оператор присвоювання (=), після цього значення, що присвоюється. Оператор присвоювання неприпустимо виділяти пробілами. Змінній можна привласнити будь-яку послідовність символів. У наступному прикладі змінній **greeting** привласнюється стрічкове значення **"Hello".**

```
$ greeting="Hello"
```

Після присвоєння значення можна користуватися ім'ям змінної для посилання на це значення, наприклад використовувати його як аргумент команд сценарію. На значення змінної можна посилатися за допомогою її імені, перед яким ставиться оператор **$**. Знак долара — це спеціальний оператор, який використовує ім'я змінної для посилання на її значення, тобто фактично для її обчислення.

1.3 Команда let

Команда **let** — це команда інтерпретатора BASH shell, яка забезпечує виконання операцій над арифметичними величинами. За допомогою цієї команди можна порівнювати числові значення чи виконувати над ними арифметичні операції. Необхідність в арифметичних обчисленнях нерідко виникає при вирішенні задач адміністрування і виконанні необхідних обчислень у керуючих конструкціях shell-сценаріїв. Команда **let** задається за допомогою ключового слова **let**. Базовий формат команди включає ключове слово **let**, за яким слідують два числових значення, розділених арифметичним оператором або оператором порівняння:

```
$ let значення1 оператор значення2
```

У shell-сценаріях може використовуватися будь-який з перерахованих у табл. 1 операторів.

Таблиця 1. Оператори BASH shell

Арифметичні оператори	**Функції**
*	Множення
/	Ділення
+	Додавання
(	Віднімання
%	Ділення з остачею
Оператори порівняння	**Функції**
>	Більше ніж
<	Менше ніж
>=	Більше або рівно
<=	Менше або рівно
=	Рівність у виразах
= =	Рівність в команді **let**
!=	Не рівно
&	Логічне І
|	Логічне АБО
!	Логічне НЕ

У процесі виконання команда **let** автоматично обчислює всі змінні і перетворить їх значення в арифметичні величини. Це дозволяє записувати всі арифметичні операції у виді простих арифметичних виразів. У наступному прикладі команда **let** використовується для перемноження двох чисел — 2 і 7. Результат поміщається в стандартний потік виводу і відображається на екрані.

1.4 Команда test

При написанні сценаріїв нерідко застосовуються керуючі конструкції, що виконують порівняння двох чи декількох величин. Порівнювати значення можна не лише за допомогою умовних керуючих конструкцій, але і за допомогою спеціальної команди Linux - **test**. Ця команда виконує порівняння двох значень, повертаючи код завершення 0 у тому випадку, якщо порівняння було успішним.

Команда **test** дозволяє порівнювати цілі числа, рядки і навіть виконувати логічні операції. Синтаксис команди такий: ключове слово **test**, за ним слідують порівнювані величини, розділені опцією, що задає тип порівняння.

```
test значення -опція значення
test рядок = рядок
```

Опцію можна розглядати як спеціальний оператор, але вона записується як звичайні опції, буквеним кодом зі знаком «мінус». Однак існують дві строкові операції, у яких замість опції дійсно використовується оператор. При виконанні порівняння двох рядків застосовується знак рівності =. Для встановлення факту нерівності двох рядків використовується знак **!=**. У табл. 2 перераховані всі опції й оператори, які використовуються командою **test**.

Таблиця 2. Операції, які виконуються командою **test**

Порівняння цілих чисел	**Функція**
-gt	Більше ніж
-lt	Менше ніж
-ge	Більше ніж або рівно
-le	Менше ніж або рівно
-eq	Рівно
-ne	Не рівно
Порівняння рядків	**Функція**
-z	Перевірка на наявність порожнього рядка
-n	Перевірка на наявність стрічкового значення
=	Перевірка рядків на рівність
! =	Перевірка рядків на нерівність
str	Перевірка на наявність рядка, що складається з нулів
Логічні операції	**Функція**
-a	Логічне І
-o	Логічне АБО
!	Логічне НЕ
Перевірка файлів	**Функція**
-f	Установка факту існування файлу
-s	Перевіряється, чи не є файл порожнім
-r	Перевірка можливості зчитування з файлу
-w	Перевірка можливості запису у файл, а також його зміни
-x	Перевіряється, чи є файл виконавчим
-d	Перевіряється, чи є ім'я файлу ім'ям каталогу
-h	Перевіряється, чи є ім'я файлу символічним посиланням

1.5 Керуючі конструкції

Існує два види керуючих конструкцій: *циклічні (цикли)* і *умовні (умови)*. Циклічна конструкція використовується для повторного виконання команд, тоді як умовна — для виконання послідовності команд, яка задовольняє визначеним умовам. В інтерпретаторі BASH shell можна використовувати три циклічні конструкції: **while**, **for** і **for-in**, і дві умовні — **if** і **case**.

Керуючі конструкції **while** і **if** — це конструкції загального призначення, що використовуються при рішенні таких задач, як ітераційні обчислення і перевірка різних умов. Керуючі конструкції **case** і **for** орієнтовані на більш вузьке коло задач. Конструкція **case** є багатоваріантним оператором і являє собою окремий випадок умовного оператора **if**. Ця конструкція часто використовується при створенні меню. Конструкція **for** являє собою цикл, що однократно обробляє всю інформацію для кожного значення, включеного в список, доти, поки не зустрінеться закінчення списку.

Крім порівняння значень чи змінних, керуючі конструкції **if** і **while** можна застосовувати для перевірки того, успішно чи невдало була виконана системна команда Linux. Нагадаємо, що в Linux кожна виконувана команда повертає код завершення. Якщо виконання команди було успішним, її код завершення дорівнює 0. Якщо з будь-якої причини команда не була виконана успішно, кодом завершення буде деяке позитивне значення, що вказує тип помилки. Керуючі конструкції **if** і **while** дозволяють перевірити, чому був рівний код завершення: 0 чи деякому іншому значенню.

1.6 Умови

Команда **if** починається ключовим словом **if**, за ним зазначена команда Linux, код завершення якої перевіряється. Ця команда виконується в будь-якому випадку. Потім в окремому рядку випливає ключове слово **then**. Далі може йти будь-яка послідовність команд. Ключове слово **fi** завершує команду.

Іноді потрібно вибрати один із двох варіантів, керуючись тим, успішно чи ні була виконана команда Linux. Ключове слово **else** конструкції **if** дозволяє вибрати один із двох варіантів. Якщо виконання команди Linux було успішним, викликаються команди, що слідують за ключовим словом **then**. Якщо ж виконання команди Linux завершилося невдачею, активізуються ті команди, що слідують за ключовим словом **else**.

Таблиця 1. Умовні керуючі конструкції інтерпретатора BASH shell

Умовні керуючі конструкції	Функція
if *команда Linux* **then** *команди* **fi**	Конструкція **if** викликає виконання команд у випадку, якщо результат перевірки виконання команди Linux істинний
if *команда Linux* **then** *команди 1* **else** *команди 2* **fi**	Конструкція **if-else** викликає виконання *команд 1* у випадку, якщо код завершення команди Linux, що перевіряється, дорівнює значенню “істина”, у противному випадку виконується *команди 2* після ключового слова **else**
if *команда Linux* **then** *команди 1* **elif** *команди 2* **then** *команди 3* **else** *команди 4* **fi**	Конструкція **elif** дає можливість вкладати конструкції **if**, що дозволяє вибрати один з багатьох варіантів; якщо істинна умова, яка перевіряється конструкцією **if**, то виконуються передбачені в ній команди 1 і наступній конструкції **elif** керування не передається. В іншому випадку, якщо умова фальшива, то виконуються команди конструкції **elif**.
Case *рядок* **in** *шаблон)* *команди* *;;* *шаблон)* *команди* *;;* **esac**	Конструкція **case** порівнює стрічкове значення з одним із декількох шаблонів (зразків). При виявленні збігу виконуються команди, що відповідають цьому шаблону.
Команда **&&** *команда*	Логічна операція І повертає значення 0 (“істина”), якщо обидві команди повертають значення 0; якщо ж одна з команд повертає ненульове значення, результат операції І дорівнює “фальш” і дана

	операція повертає ненульове значення
команда **||** *команда*	Логічна операція АБО повертає значення 0 (“істина”), якщо одна чи обидві команди повертають значення 0 («істина»); якщо обидві команди повертають ненульове значення, то результат операції АБО — “фальш” і операція повертає ненульове значення
! *команда*	Логічна операція НЕ, інвертує код завершення команди

Конструкція **elif** дозволяє виконати вкладення операції **if-then-else**. Абревіатура **elif** позначає **else if**. За допомогою **elif** можна вибирати один з декількох варіантів. Перший варіант задається конструкцією **if**. Далі вказуються інші варіанти, кожному з який відповідає власна конструкція **elif**. Варіант для останньої конструкції **elif** задається ключовим словом **else**. Якщо перевірка за допомогою першої конструкції **if** дає негативний результат, керування передається наступній конструкції **elif** і виконується передбачена там перевірка. Якщо й у ході цієї перевірки отримано негативний результат, керування передається наступному оператору **elif**, після чого знову виконується перевірка. Процес продовжується доти, поки результат перевірки не стане істинним. Конструкція **elif** має власні виконувані команди, після завершення яких керування передається команді, що слідує після ключового слова **fi**.

Конструкція **case** забезпечує вибір одного з декількох альтернативних варіантів. Вибір здійснюється шляхом порівняння □ргумент в конструкції значення з декількома можливими шаблонами. Кожне можливе значення змінної, що перевіряється зв'язується з набором операцій. Якщо встановлюється збіг, виконуються відповідні □ргумен значенню операції. Конструкція **case** починається ключовим словом **case**, за яким слідує ім'я порівнюваної змінний і ключове слово **in**. Далі перелічується набір шаблонів. Кожен шаблон являє собою вираз, який закінчується закриваючою круглою дужкою. Після круглої дужки, слідують команди, зв'язані з цим шаблоном. Список команд завершується двома символами “крапка з комою” в окремому рядку, що вказують на завершення команд. Усю конструкцію завершує ключове слово **esac**. Шаблон може включати будь-які спеціальні символи інтерпретатора команд (***, [ ], ? і |**).

Логічні команди призначені для виконання логічних операцій із двома командами Linux. Логічна операція І (**&&**) вважається успішною, якщо результати виконання обох команд успішні. У випадку логічного АБО (| |) операція вважається успішною і видає код завершення 0, якщо результат виконання однієї з команд успішний. Логічні операції використовуються при заданні умов перевірки в керуючих конструкціях.

1.7 Цикли

В інтерпретаторі BASH shell використовується набір керуючих циклічних конструкцій, які забезпечують повторне виконання команд Linux. Керуючі циклічні конструкції перераховані в табл. 2.

Таблиця 2. Циклічні керуючі конструкції інтерпретатора BASH shell

Циклічні керуючі конструкції	**Функція**
while *команда Linux* **do** *команди* **done**	Конструкція **while** виконує дію до тих пір, поки команда перевірки повертає значення “істина”
until *команда* **do** *команди* **done**	Конструкція **until** виконує дію до тих пір, поки команда перевірки повертає значення “фальш”
for *змінна* **in** *список-значень* **do** *команди* **done**	Конструкція **for-in** призначена для обробки списку значень. Змінній послідовно привласнюються значення зі списку
for *змінна* **do** *команди* **done**	Конструкція **for** призначена для послідовної обробки аргументів сценарію. Змінній послідовно привласнюється значення кожного аргументу
select *рядок* **in** *переліків-елементів* **do** *команди* **done**	Конструкція **select** створює меню на основі елементів □ргумент списку, а потім виконується зазначена команда (зазвичай це команда **case**)

Цикл **while** використовується для повторення команд. Цей цикл починається ключовим словом **while**, за яким йде команда Linux. У наступному рядку повинно стояти ключове слово **do**. Закінчення циклу позначається ключовим словом **done**.

Конструкція **for** використовує в якості списку значень аргументи командного рядка. Списком значень, до якого звертається команда **for**, стають аргументи, що вказуються в командному рядку при виклику файлу shell. Всі ргумент из цього списку автоматично привласнюються змінній, що використовується в команді **for**. При першій ітерації циклу змінної привласнюється значення першого аргументу, при другий — другого і т.д.

Засоби та підготовка:

Логін та пароль до робочого місця з компютером, що працює під ОС Linux.

Порядок виконання роботи:

5.Запустити систему та залогуватись.

6.Прочитати скріпт та дописати коментарі до кожного рядка.

```
Echo “Main menu”
echo « [P]rint a file»
echo « [D]elete a file»
echo « [Q]uit»
read response
case $response in
      P|p) echo «Name of file to print?»
          read filename
          lp $filename;;
      D|d) echo «Name of file to delete?»
          read filename
          rm $filename;;
      Q|q) echo «Exit now»
          exit;;
        *) echo “Unrecognized option”;;
   Esac
```

3. Написати власний скріпт, котрий в діалоговому режимі проводить правильне копіювання файлу 1 у файл 2 з перевіркою на можливість читання файлу 1 та можливість запису у файл 2 а також існування файлу 2. При неправильному використанні параметрів скріпта видавати допомогу по формату задання його параметрів.

4. Задати скріпту атрибут виконавчого файлу та перевірити скріпт в роботі.

5. Завершити роботу системи.

3. Порядок оформлення звіту по роботі.

3.1 На титульній сторінці креслярським шрифтом вказати назву інституту, кафедри, назву і номер роботи, прізвище, ініціали, номер групи виконавця, прізвище і ініціали викладача, який керував роботою, рік виконання роботи.

3.2 Вказати мету роботи.

3.3 Привести завдання для розробки.

4. Контрольні запитання.

4.1 Який оператор використовується для математичних обчислень ?

4.2 Якою командою можна записати значення у змінну та зчитати його ?

4.3 Який синтаксис оператора вибору case?

4.4 Якою командою можна вивести повідомлення на екран?

Лабороторна робота №7

Тема: Робота з програмними пакетами, створення облікового запису користувача
Мета: Проводити встановлення та видалення програмних пакетів та їх аналіз.
Навчитись створювати і модернізовувати обліковий запис користувача.

1. Підготовчі відомості:

1.1 Введення в керування пакетами програм

Система керування пакетами програм є частиною операційної системи Linux, яка дозволяє встановлювати і видаляти програмне забезпечення на комп'ютері користувача і працювати з ним.

Майже всі дистрибутиви операційної системи Linux містять у собі різні системи керування пакетами програм. Найбільш розповсюдженою з них є ***Red Hat package manager (RPM)***. RPM дає можливість користувачу виконувати наступні задачі:

- досліджувати вміст інсталяційного пакета і визначити обсяг дискового простору, необхідного для його установки, а також з'ясувати, які файли входять до складу встановлюваної програми і де саме у файловій системі вони будуть розташовані;
- перевірити наявність усіх необхідних бібліотек чи будь-яких файлів, що вимагаються для роботи нового пакета програм;
- визначити, чи буде дана програма конфліктувати із раніше встановленими програмами;
- досліджувати процедуру інсталяції і визначити, чи не порушується при цьому цілісність операційної системи і системного захисту;
- переконатися, що пакет отриманий з надійного і перевіреного джерела;
- перевірити, чи не були файли пакета ушкоджені чи видалені;
- з'ясувати всю необхідну інформацію, що відноситься до даного пакета і стосується, зокрема, першоджерела програмного забезпечення, а також існуючих обмежень на його використання і тиражування;
- обновити раніше встановлені програми при збереженні всіх існуючих настроювань у файлах конфігурації даного програмного забезпечення;
- визначити, якому пакету належить той чи інший файл;

Пакет являє собою спеціальний файл, що містить у собі каталоги і файли, які, в свою чергу, є частиною окремого фрагмента програмного забезпечення чи системного компонента. Крім того, у пакеті міститься додаткова інформація, що відноситься до даного програмного забезпечення, яка використовується диспетчером пакетів. Пакети бувають двох видів: двійкові (бінарні) і пакети вихідних текстів. Двійковий (binary) пакет містить програмні компоненти, сформовані для визначеної архітектури і цілком підготовлені до використання. Пакет вихідних текстів (source) містить у собі вихідний програмний код і інші елементи, які використовуються для формування відповідного двійкового пакета. Пакети вихідних текстів дозволяють змінити програмний код і, отже, саму програму.

Імена файлів двійкового пакета мають розширення . rpm, на відміну від них імена файлів пакетів вихідних текстів закінчуються суфіксом. src. rpm.

Ім'я файлу двійкового пакета виглядає приблизно так:

ed-0.2-5.i386.rpm

Першою частиною імені файлу є коротке слово (ім'я), що ідентифікує пакет (у даному випадку, ed). Після імені йде число, що визначає номер версії програмного забезпечення, яке знаходить у пакеті (версія 0.2). За цим номером йде номер випуску пакета (5). Варто розрізняти версію програмного забезпечення і випуск пакета: номер версії являє собою версію програмного забезпечення, що знаходиться в даному пакеті. У прикладі, це програма ed версії 0.2. Номер версії відбиває стан програмного забезпечення. Номер випуску відноситься до версії саме цього пакета. Іноді навіть якщо програмне забезпечення, що знаходиться усередині пакета практично не змінилося, необхідно перепакувати його. Таким чином, номер випуску вказує номер існуючого варіанта пакета. За номером випуску слідує коротке слово, що вказує архітектуру пакета. Найчастіше пакети містять у собі програмне забезпечення, що було відкомпільоване для визначеного процесора чи визначеної архітектури комп'ютера. Наприклад, в імені пакета, програми якого були відкомпільовані для архітектури Intel x86 чи сумісних процесорів, зазвичай знаходиться рядок архітектури i386. У тому випадку, якщо пакетні програми спеціально відкомпільовані для процесорів Pentium чи Pentium II, в імені пакета буде присутній рядок i586 чи i686, відповідно. Іноді зміст пакета зовсім не залежить від архітектури комп'ютера (наприклад, пакет містить лише піктограми і не містить виконуваних програм). У цих випадках рядок архітектури містить слово **noarch**.

Двійковий пакет містить наступні елементи:

- ім'я і короткий опис програмного продукту;
- номер версії;
- дані, що відносяться до постачальника RPM;
- URL-адреса головної Web-сторінки даної програми;
- інформацію про ліцензію й авторські права;
- класифікацію програмного продукту;
- операційну систему і дані про архітектуру програми;
- дані про кожен з файлів і каталогів, що входять до складу пакета (сюди входять імена файлів, розміри, права доступу, контрольні суми, класифікація файлів: звичайні, конфігураційні чи файли документації);
- інформацію про переміщення, що дозволяє установити пакет програм у різні каталоги;
- програмні вимоги і відомості про можливі конфлікти з іншими пакетами, операційними системами й архітектурами;
- попередні версії даного пакета;
- сценарії, які виконуються при інсталяції і видаленні програми;
- інформацію про перевірку цілісності і працездатності пакета до і після його інсталяції;
- список останніх змін.

1.2 Робота з rpm

Для керування пакетами програм використовується команда **rpm**. Основні параметри команди **rpm** приведені в табл. 1.

Таблиця 1. Основні параметри команди rpm

Параметри	Підпараметри	Значення
-q		Запит
	-i	Одержання докладної інформації про пакет
	-l	Перелік усіх файлів
	-d	Відображення тільки файлів документації
	-c	Відображення тільки файлів конфігурації
	-f	Пошук пакета, якому належить даний файл
	-p	Виконує дії з файлами пакета (який ще не встановлений)
	--scripts	Перегляд сценаріїв інсталяції/видалення
-i		Установка (інсталяція)
-e		Видалення (деінсталяція)
-U		Відновлення чи інсталяція пакета, якщо він ще не встановлений
-F		Відновлення (це відноситься лише до раніше встановленого пакета)
-V		Перевірка інсталяції пакета
-b		Формування пакета

1.3 Отримання інформації про пакети

Для того, щоб одержати інформацію, що відноситься до певного пакету, потрібно виконати наступну команду:

rpm -qpi ed-0.2-5.1386.rpm

На екрані з'явиться перелік параметрів пакету.

Для того щоб побачити перелік файлів, що знаходяться в цьому пакеті, потрібно ввести команду:

rpm -qpl ed-0.2-5.1386.rpm

Щоб з'ясувати чи інстальований у системі пакет, потрібно ввести наступну команду:

rpm -qi ed

При виконанні даної команди буде виведена дата інсталяції пакета. У тому випадку якщо пакет не встановлений, **rpm** повідомить про це.

Для того щоб запросити будь-яку інформацію про пакет чи встановити його, користувачу необхідно звернутися безпосередньо до файлу пакета, вказавши при цьому його повне ім'я (наприклад, ed-0.2-5.i386.rpm). Це відноситься до неінстальованого програмного забезпечення. Для виконання будь-яких дій з пакетами, встановленими в системі (наприклад, запити, перевірка чи видалення), варто вказати лише ім'я пакету (наприклад, ed).

1.4 Встановлення програмного забезпечення

Для інсталяції програмного забезпечення потрібно ввійти в систему як суперкористувач, і ввести команду **rpm** з опцією **-i** й ім'ям файлу пакета, який потрібно встановити:

 # rpm -i /mnt/cdrom/Packages/RPMS/ytalk-3.1-1.i386.rpm

Для того щоб відстежити розвиток інсталяції, можна ввести команду **rpm** з опціями **vh**:

#rpm -ivh /mnt/cdrom/Packages/RPMS/ytalk-3.1-1.i386.rpm

Для повнішого звіту про хід інсталяції, можна скористатись опцією **vvh**:

#rpm -ivvh /mnt/cdrom/Packages/RPMS/ytalk-3.1-1.i386.rpm

1.5. Видалення пакета

Так само, як і при інсталяції програм, для виконання операцій видалення потрібно ввійти в систему як суперкористувач. Для видалення пакета із системи необхідно ввести команду **rpm** з опцією **-e** й ім'ям пакету, який потрібно видалити:

 # rpm -e ytalk

1.6. Відновлення пакета

Програма RPM може бути використана для відновлення пакета, що існує у системі, для установки більш пізнього випуску пакета чи ж більш нової версії програмного забезпечення. Логічно відновлення пакета являє собою видалення пакета, раніше встановленого в системі, і наступну інсталяцію нового. Для оновлення пакета, що існує в системі, потрібно скористатись командою rpm з опцією –U. Наприклад, користувач встановив в системі пакет perl-5.5.3, а тепер хоче обновити його до perl-5.6.0. Для цього потрібно ввійти в систему як суперкористувач і виконати операцію відновлення:

 # rpm -U perl-5.6.0-l.i386.rpm

1.7. Перевірка пакетів

За допомогою RPM можна перевірити коректність файлів конфігурації, а також з'ясувати, чи виконані всі необхідні операції для установки даного пакета. Для того щоб перевірити пакет, потрібно скористатись командою **rpm** з опцією -V:

$rpm -V apache

1.8 Типи облікових записів користувачів

Оскільки Linux — багатокористувацька система, то додавання нових і супровід існуючих **облікових записів користувачів** (**user accounts**) є звичайною задачею системного адміністрування Linux. Після успішної інсталяції дистрибутива Linux автоматично створюються два облікові записи: для користувача root (суперкористувача) і для звичайного користувача. Ці два облікові записи користувачів являють собою два базових типи користувачів, які існують у Linux.

Перший тип — користувач **root** (суперкористувач чи superuser)— це єдиний обліковий запис користувача з необмеженим доступом до всіх ресурсів системи. Обліковий запис суперкористувача ретельно захищений. Для безпеки системи надзвичайно важливий пароль облікового запису суперкористувача. Пароль суперкористувача необхідно давати тільки тим, хто уповноважений реєструватися в системі під ім'ям root, і періодично його необхідно змінювати. Тільки обліковий запис root використовується для виконання задач, пов'язаних із системним адмініструванням, чи задач, що можуть бути виконані тільки суперкористувачем. Найкраще виконувати як можна більше задач в якості звичайного користувача і лише у випадку крайньої необхідності використовувати обліковий запис суперкористувача.

Обліковий запис користувача root є не єдиним спеціальним обліковим записом, що існує в системі. В інстальованому дистрибутиві Linux можна побачити такі облікові записи, як bin, daemon, adm, lp, sync, shutdown, mail, operator і інші. Усі вони також є спеціальними обліковими записами. Вони називаються системними обліковими записами і використовуються для різних цілей. Ці облікові записи не мають привілеїв, наданих користувачу root, і для запобігання потенційних проблем, пов'язаних із захистом системи, визначені додатки запускаються від імені даних користувачів, а не від імені суперкористувача. У системних облікових записів відсутні паролі, оскільки вони не призначені для реєстрації в системі. Однак необхідно, щоб ці записи були присутні у файлі паролів /etc/passwd. Їх не можна вилучати з файлу паролів, оскільки деякі програми, при цьому, не будуть працювати. Ці спеціалізовані облікові записи також називаються обліковими записами без реєстрації.

1.8.1 Конфігураційні файли облікових записів користувачів При створенні нового облікового запису користувача використовуються визначені стандартні файли, названі конфігураційними та каталоги. Існує набір шляхових імен, які визначають їх розташування. Перелік шляхових імен:

Каталог	Опис
/home	Місцезнаходження початкового каталогу користувача
/etc/passwd	Містить інформацію про всіх користувачів системи
/etc/shadow	Містить інформацію про тіньові паролі користувачів

1.8.2 Конфігураційний файл /etc/passwd

Ключовим файлом при настойці і конфігурації облікових записів користувачів є файл **/etc/passwd**. Це звичайний текстовий файл у форматі ASCII, який відіграє важливу роль у багатокористувацькій концепції Linux. Після успішної інсталяції системи файл **/etc/passwd** містить приблизно наступну інформацію:

```
root:x:0:0 rroot:/root:/bin/bash
bin:x:l:l:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
ftp:X:14:50:FTP User:/home/ftp:
petro:x:101:100:OpenLinux User:/home/petro:/bin/bash
olga:x:102:100:OpenLinux User:/home/olga:/bin/bash
```

Кожен рядок файлу являє собою запис про одного користувача. Інформація про кожного користувача знаходиться в сімох, розділених двокрапкою, полях. Опис кожного з цих полів приведено в табл. 1.

Таблиця 1. Опис полів файлу **/etc/passwd**

Поле	Опис
1	Зберігається ім'я користувача чи ідентифікатор облікового запису. Це поле повинне бути унікальним
2	Поле паролю. Символ “x” означає, що в системі використовуються тіньові паролі. Реальні паролі для забезпечення безпеки зберігаються в зашифрованому виді в окремому файлі
3	Число в цьому полі позначає ідентифікатор користувача (User ID, UID). Він використовується для ідентифікації облікового запису. Це число повинне бути унікальним
4	Ідентифікатор групи (Group ID — GID). Дане число використовується для створення груп користувачів. Тут вказується група, в яку входить користувач
5	Це поле для коментарів. Зазвичай в ньому міститься повне ім'я користувача
6	Використовується для задання робочого каталогу, у якому користувач з’явиться після реєстрації в системі
7	Використовується для задання оболонки по замовчуванню.

Облікові записи нових користувачів можуть бути додані шляхом безпосереднього редагування файлу **/etc/passwd**, але для створення паролів необхідно скористатися утилітою **/usr/bin/passwd**. Ця утиліта потрібна для здійснення правильного шифрування паролю.

1.8.3 Конфігураційний файл /etc /shadow

Доступ до будь-якого облікового запису контролюється за допомогою паролю. У більшості дистрибутивів Linux по замовчуванню використовуються тіньові паролі.

Тіньові паролі - це паролі, які у зашифрованому виді зберігаються в окремому файлі. Файл **/etc /shadow** забезпечує додатковий рівень безпеки системи.

До того, як з'явилися тіньові паролі, зашифрований пароль зберігався у файлі **/etc/passwd**. Вміст файлу **/etc/passwd** доступний для читання будь-якому користувачу системи, але змінювати його вміст може тільки суперкористувач. Цей файл повинен бути доступний для читання всім користувачам, тому що інакше деякі додатки не зможуть працювати. Однак це являє визначену проблему, оскільки кожен може побачити зашифрований пароль. Поява потужних комп'ютерів привела до того, що стало набагато простіше зламати навіть відмінно обрані паролі. Для вирішення даної потенційної загрози була розроблена концепція тіньових паролів. Зберігаючи паролі в окремому файлі **/etc/shadow**, стало можливим залишити файл **/etc/passwd** доступним для читання для всіх облікових записів, але самі зашифровані паролі зробити доступними для читання тільки для користувача root.

Вміст файлу **/etc/shadow** приблизно наступний:

```
root:Rb3mAtGdwFMDE:10568:0:-1:7:7 :-1:1073897392
bin:*:11542:0::7:7::
daemon:*: 11542:0::7:7::
adm*:11542:0::7:7::
ftp:*:11542:0::7:7::
petro:jEFpYlZqmlSE.:11576:0:-1:9:-1:-1:1073897392
olga:hBcIkjWnhFk/o:11576:0:-1:7;-1:-1:1073897392
```

Подібно файлу **/etc/passwd** файл **/etc/shadow** складається з рядків із полями, розділених двокрапкою. Деякі з цих полів містять числа, які прив'язані до дати початку "епохи обчислювального століття" — 1 січня 1970 року. Опис кожного з полів приведено в табл. 2.

Таблиця 2. Опис полів файлу **/etc/shadow**

Поле	Опис
1	Ім'я облікового запису
2	Поле, що містить зашифрований пароль
3	Кількість днів, що пройшли з початку епохи, коли пароль востаннє мінявся
4	Кількість днів, які повинні пройти до того, як потрібно буде знову змінити пароль
5	Кількість днів, після яких пароль повинен бути змінений (значення -1 означає, що ніколи)
6	Кількість днів, що залишилися до того, як пароль перестане діяти, і протягом яких користувач буде одержувати повідомлення про це при реєстрації в системі
7	Кількість днів після закінчення терміну дії пароля, протягом яких обліковий запис буде блоковано, якщо пароль не буде успішно змінений
8	Кількість днів, що пройшли з початку епохи, після якого обліковий запис буде заблоковано
9	Зарезервоване поле

1.8.3 Управління обліковими записами користувачів

Керувати обліковими записами користувачів можна безпосередньо редагуючи файл **/etc/passwd** чи використовуючи утиліти **adduser**, **useradd**, **usermod** і **userdel**, які є у більшості дистрибутивів ОС Linux. Усі ці утиліти одержують інформацію з командного рядка у виді параметрів.

При добавленні в систему нового користувача можна використати команду **adduser**. Дана команда використовує в якості параметра ім’я користувача створюваного облікового запису.

 # adduser larisa

Після натиску клавіші [Enter], буде створений новий обліковий запис з використанням стандартних значень. Після створення реєстраційного імені нового користувача потрібно задати для нього пароль. Для задання паролю для нового облікового запису можна застосувати команду **passwd**.

Утиліта **useradd** при створенні нового облікового запису використовує набір наперед визначених стандартних значень, у тому числі ім'я групи, ідентифікатор користувача, початковий каталог, каталог **skel** і реєстраційний **shell**. Ім'я вказується для групи, до якої належить новий обліковий запис (по замовчуванню записано *other*). Дане ім'я означає, що новий обліковий запис не належить ні до однієї групи. Ідентифікатори служать для позначення облікових записів користувачів. Перший обліковий запис має ідентифікатор 1, другий — 2 і т.д. Каталог **skel** — це системний каталог, що містить копії файлів ініціалізації. Ці файли копіюються в новий початковий каталог користувача при його створенні. Під реєстраційним shell розуміють шляхове ім'я shell, з яким користувач буде працювати. Стандартні значення можна одержати за допомогою команди **useradd -D**.

Команда **useradd** має опції, що дозволяють змінювати стандартні значення (табл. 3). Після введення у систему нового реєстраційного імені користувача необхідно привласнити йому пароль. У наступному прикладі ім'я групи облікового запису olga — intro, а ідентифікатор користувача — 578.

 # useradd olga -g introl -u 578

Таблиця 3. Команди для роботи з обліковими записами користувачів

Команда	Опис
adduser *ім'я_користувача*	Вводить нового користувача, створюючи запис у файлі паролей, а також початковий каталог і каталог пошти з файлами ініціалізації. Використовує команду **passwd** для задання пароля користувача.
useradd *ім'я_користувача опції*	Вводить у систему облікові записи нових користувачів
usermod *ім'я_користувача опції*	Модифікує облікові записи користувача
userdel *ім'я_користувача опції*	Видаляє із системи обліковий запис зазначеного користувача
Опції команд useradd, usermod	
-u *ідентифікатор_користувача*	Встановлює ідентифікатор нового користувача; по замовчанню збільшується на одиницю самий більший номер з чисел що застосовувалися
-g *група*	Включає користувача в зазначену групу
-d *каталог*	Задає початковий каталог нового користувача
-s *shell*	Задає реєстраційний shell нового користувача
-c рядок	Додає коментар у запис, що відноситься до даного користувача і знаходиться у файлі паролів **/etc/passwd**

-D	Видає значення за замовчуванням для всіх параметрів облікового запису користувача

Команда **usermod** дозволяє змінювати значення цих елементів. Можна, наприклад, змінити початковий каталог, ідентифікатор користувача і навіть ім'я облікового запису.

Видалення чи блокування облікових записів користувачів здійснюється за допомогою редагування файлу **/etc/passwd**. Наприклад, для блокування облікового запису користувача petro, потрібно змінити значення в полі паролю наступним чином

petro:xl23:100:100:0penLinuxUser:/home/natalie:/bin/bash

Коли до символу x приписується ще що-небудь (у даному випадку символи 123), то утиліти, які працюють з тіньовими паролями, більше не зможуть звертатися до файлу **/etc/shadow** для одержання пароля користувача, а замість цього в якості паролю спробують використовувати значення з поля паролю у файлі паролів. Оскільки жоден пароль не зможе бути зашифрований так, щоб збігатися з даним значенням у полі пароля (x123), то цей обліковий запис не зможе використовуватися. Зміна значення поля знову на символ x знову активізує обліковий запис користувача.

Повне видалення запису з файлу **/etc/passwd** також приведе до блокування облікового запису користувача, але якщо треба буде знову її відновити, то зробити це буде складніше.

Для того, щоб видалити із системи обліковий запис користувача, можна також скористатись утилітою **userdel**. У приведеному прикладі, із системи видаляється обліковий запис користувача olga.

 # userdel -r olga

1.9 Керування обліковими записами груп

1.9.1 Робота з обліковими записами груп

Ідентифікатор групи GID призначений для логічного групування ресурсів і файлів з метою їхнього спільного використання членами даної групи. Наприклад, файли з вихідними кодами можуть спільно використовуватися співробітниками конструкторського відділу. Створивши групу **eng** і включивши в її склад необхідних користувачів, співробітники конструкторського відділу зможуть спільно використовувати ці файли, а інші користувачі не будуть мати до них доступу.

1.9.2 Створення групи

Системний файл, у якому знаходяться елементи облікових записів груп, називається **/etc/group**. Вміст цього файлу приблизно наступний:

```
root::0:
bin::1:
daemon::2:bin,daemon
ftp::50:
users::200:
```

Для кожної групи файл **/etc/group** містить записи у вигляді рядків, поля яких розділені двокрапками. Опис кожного з цих полів приведено в табл. 1.

Таблиця 1. Опис полів файлу **/etc/group**

Поле	Опис
1	Ім'я групи. Цей параметр повинен бути унікальним
2	Поле пароля для групи. Зазвичай він не використовується, але в принципі може бути встановлений
3	Ідентифікатор групи.
4	Список облікових записів користувачів, що входять до складу цієї групи

Для створення нової групи необхідно додати запис про неї у файл **/etc/group**. Нижче приведений приклад запису з файлу **/etc/group**. Група називається **eng**, пароля немає, ідентифікатор групи — 100, у групу входять користувачі **petro**, **olga** і **olena**.

eng::100:petro,olga,olena

Для управління обліковими записами груп також можна застосовувати команди **groupadd**, **groupmod** і **groupdel** (табл.2)**.**

Команда **groupadd** призначена для створення нових груп. Нова група реєструється у файлі **/etc/group** і отримує ідентифікатор. Спочатку група, створена командою **groupadd**, є порожньою. Користувачі вводяться в групу по черзі. У наступному прикладі команда **groupadd** застосовується для створення групи **eng**.

```
# groupadd engines
```

1.9.3 Додавання користувачів у групу

Користувач може входити до складу необмеженої кількості груп. Для того щоб включити користувача

до складу якоїсь групи у файлі **/etc/group** необхідно додати ім'я даного облікового запису в поле зі списком імен облікових записів, що входять у групу. Після додавання облікового запису користувача він одержує доступ до файлів, що спільно використовуються цією групою.

Для прикладу, щоб додати користувача **taras** у групу **eng**, потрібно змінити запис наступним чином:

eng::100:petro,olga,olena,taras

Таблиця 2. Команди для роботи з обліковими записами груп

Команди управління обліковими записами груп	**Опис**
groupadd	Створює нову групу
groupdel	Видаляє групу
groupmod *опція*	Модифікує обліковий запис групи: **-g** — змінює ідентифікатор групи **-n** — змінює ім'я групи

Існує також корисна утиліта **gpasswd**, що входить до складу набору утиліт, пов'язаних із роботою з тіньовими паролями, яка поставляється разом з Linux. Для включення нового користувача до складу групи ця утиліта використовується з наступним синтаксисом:

gpasswd -a *<ім'я користувача> <група>*

Для одержання списку всіх груп, до складу яких входить користувач використовується команда **groups** . Для того щоб одержати цей список для конкретного користувача, потрібно виконати команду **groups** і в якості параметру вказати ім'я його облікового запису.

1.9.4 Видалення користувачів із групи

Для того щоб вивести користувача зі складу якоїсь групи, у файлі **/etc/group** необхідно видалити ім'я користувача зі списку облікових записів користувачів, що входять до складу цієї групи. Це не спричинить за собою видалення даних чи зміни прав доступу — просто користувач позбавиться доступу до файлів, якими володіє ця група.

Щоб видалити користувача зі складу групи, можна скористатись також утилітою **gpasswd з** наступним синтаксисом:

gpasswd -d *<ім'я користувача> <група>*

Видалити групу дозволяє команда **groupdel**. Наприклад, якщо видаленню підлягає група **eng**, потрібно виконати наступну команду:

```
# groupdel eng
```

Призначення команди **groupmod** — зміна імені групи і її ідентифікатора. Щоб зробити цю операцію, потрібно ввести команду **groupraod -g** з новим ідентифікатором і ім'ям групи. Для заміни імені групи служить опція **-n**. У наступному прикладі показано, як групу eng перейменувати в trains.

```
$ groupmod -n trains eng
```

2 Порядок виконання роботи:

Запустити систему та залогуватись.

2.1 RPM пакети

Переглянути список встановлених пакетів та їх властивості.

а) переглянути сортований список всіх встановлених пакетів rpmquery –all|sort. Короткий опис виводу на термінал.

б) вивести інформацію про пакет компілятора rpmquery –info gcc. Короткий опис виводу на термінал.

в) вивести список конфігураційних файлів та файлів документації пакету gcc.

7.Скачати з дистрибутиву пакет lynx-*.rpm.

Встановлення нових пакетів та їх видалення.

а) встановити скачаний пакет lynx.

б) перевірити його встановлення rpmquery lynx та інформацію про нього rpmquery –info lynx. Короткий опис виводу на термінал.

в) видалити пакет з системи.

г) перевірити відсутність пакету rpmverify lynx. Короткий опис виводу на термінал.

2.2 Створення облікового запису користувача

Запустити систему та залогуватись.

1. Створити користувача oleg.
2. Перевірити чи створений обліковий запис користувача.
3. Пояснити, кожне поле у файлі /etc/passwd для створеного користувача.
4. Пояснити, кожне поле у файлі /etc/shadow для створеного користувача.
5. Задати пароль для користувача.
6. Зайти в другу віртуальну консоль і залогінитись під створеним користувачем.
7. Створити обліковий запис користувача Petro.
8. Задати пароль для користувача Petro.
9. Зайти в третю віртуальну консоль і залогінитись під іменем Petro.
10. Перевірити хто в даний момент знаходиться в системі.
11. Створити групу користувачів Druzi.
12. До створеної групи Druzi додати користувачів: oleg, petro.
13. Змінити коментар користувачів oleg, petro на Druzi.
14. Перевірити застосування змін.
15. Змінити поточний каталог користувача petro на /home.
16. Перевірити застосування змін.
17. Видалити користувачів oleg, petro.
18. Перевірити застосування змін.
19. Завершити роботу системи.

Завершити роботу системи.

3. Порядок оформлення звіту по роботі.

3.1 На титульній сторінці креслярським шрифтом вказати назву інституту, кафедри, назву і номер роботи, прізвище, ініціали, номер групи виконавця, прізвище і ініціали викладача, який керував роботою, рік виконання роботи.

3.2 Вказати мету роботи.

3.3 Привести завдання для розробки.

4. Контрольні запитання.

1. Яка команда призначена для перегляду вмісту програмного пакету?
2. Якою командою можна ін сталювати пакет?
3. Якою командою деінсталюється пакет?
4. Якою командою виводиться список конфігураційних файлів?

Лабороторна робота №8

Тема: Конфігурування завантажувальника системи, компіляція ядра ОС Linux.
Мета: Проводити налаштування завантажувальника ОС LILO, навчитися компілювати ядро ОС Linux.

1. Підготовчі відомості

1.1 Адміністрування ядра

Ядро є програмою, що виконує основні функції операційної системи. Номер версії ядра ОС Linux складається з трьох компонентів — старшого і молодшого номерів, а також номера виправлень. Старший номер збільшується при внесенні істотних змін у ядро. Молодший номер дозволяє судити про стійкість версії. Непарні номери зарезервовані для стабільних версій, а парні привласнюються версіям, що перебувають у стадії розробки і можуть змінюватися. Номер виправлень змінюється після виявлення і виправлення помилок. Ядра, що знаходяться на стадії розробки, можуть неодноразово піддаватися модернізації. Приклад номера версії ядра з трьох компонентів: 2.2.16. Тут старшим і молодшим номером є 2, а номером виправлення — 16. У системах Red Hat використовується ще один номер, що ідентифікує версію доповнення ядра. Ядро операційної системи Red Hat 7.0 має номер 2.2.16-22, де 22 — це номер доповнення. У системі, що підтримує роботу з RPM-пакетами, можна довідатися, яка версія ядра встановлена, за допомогою наступної команди:

$ rpm -q kernel

Додаткові відомості про ядро ОС Linux надаються на Web-вузлі www.kernel.org — в офіційному сховищі поточних версій ядер Linux. Тут можна знайти останні версії вихідних кодів ядер, а також документацію. Установка нового ядра вимагає завантаження пакета програмного забезпечення цього ядра. Для завантаження такого пакета варто звернутися або на FTP-вузли дистрибутивів ОС Linux, або на Web-вузол www.kernel.org. Існує два способи інсталяції нового ядра:

- завантаження й установка його двійкової версії з Web-вузла дистрибутива;
- завантаження вихідного коду, його компілювання і наступна інсталяція отриманого двійкового файлу.

Процес генерування ядра складається з наступних етапів.

1. Придбання файлів вихідних кодів ядра.
2. Підготовка дерева вихідних кодів.
3. Конфігурування вихідного коду.
4. Компіляція вихідних файлів і інсталяція модулів.
5. Переміщення ядра.
6. Конфігурування і запуск LILO чи GRUB.
7. Перезавантаження системи і тестування створеного ядра.

Перед установкою нової версії варто зберегти копію поточного ядра, щоб мати можливість повернутися до нього, якщо нова версія виявиться непрацездатною. Файл ядра називається **vmlinuz*версия***, де версія — це номер версії (наприклад, **vmlinuz2.2-16**). Цей файл розміщається в каталозі **/boot**. Перед інсталяцією нової версії ядра також варто сформувати резервну копію файлу **System.map**. Цей файл містить дані, необхідні модулям для використання функцій ядра. При звертанні до ядра 2.2.16-22 цю роль відіграє файл **System.map-2.2.16-22**. Якщо потрібно після внесення ряду змін скомпілювати ту версію ядра, що уже встановлена, варто створити резервну копію модулів, розміщених у каталозі **/lib/modules/*version***, де ***version*** — це номер версії ядра. У версії 2.2.16-22 бібліотеки зберігаються в каталозі **/lib/modules/2.2.16-22**.

$ cp /boot/vmlinuz2.2.16-22 /boot/vmlinuz2.2.16-22.back

1.2 Придбання вихідних кодів ядра (етап 1)

До складу кожного дистрибутива операційної системи обов'язково входять файли вихідних кодів ядра. Випуск нової версії ядра відбувається з появою нових, чи значних змінах існуючих драйверів, а також при відновленні системи безпеки.

RPM-пакети ядра Linux можна завантажити з різних вузлів відновлення дистрибутивів. Наприклад, RPM-пакети, необхідні для завантаження нового ядра для ОС Red Hat, знаходяться на вузлі update.redhat.com або на його дзеркальних вузлах (ftp.redhat.com чи на web-вузлі www.kernel.org). В підкаталозі **/pub/linux/kernel/v2.4** потрібно знайти одну із останніх версій ядра (стиснутий архів .tar.gz) і завантажити її. Отриманий файл необхідно розпакувати в каталог **/usr/src**. Архів включає каталог з назвою **linux**, що містить файли вихідного коду ядра.

1.3 Підготовка дерева вихідних кодів (етап 2)

Дерево вихідних кодів являє собою повністю сформоване дерево каталогів і встановлених у них файлів вихідних кодів ядра і його компонентів.

Синтаксис команди, що дозволяє відновити це дерево з завантаженого файлу, залежить від формату отриманого файлу. Для прикладу, це може бути файл **linux-2.4.10.tar.gz**. Для того щоб відкрити його, потрібно виконати команду:

tar xzvf linux-2.4.10.tar.gz

При виконанні цієї команди уміст файлу розгорнеться в каталог **/usr/src/linux**. Таким чином, файли вихідного коду ядра будуть розміщені в каталозі **/usr/src/linux**. Крім того, ці файли можна знайти в каталозі **/usr/include**, який система автоматично зв'язує з каталогом **/usr/src/linux**.

1.4 Конфігурування ядра (етап 3)

На цьому етапі потрібно буде вибрати ті опції, які мають використовуватись в створюваному ядрі. Для більшості параметрів можна скористатися установками по замовчуванню. Існують три методи створення файлу конфігурації, що використовується при збиранні нового ядра:

- **make config**
- **make menuconfig**
- **make xconfig**

Вони вирішують однакові задачі, але мають різний інтерфейс. Додаток **conflg** є сценарієм конфігурування і працює в режимі командного рядка. Через велику кількість опцій і існуючі залежності однієї опції від інших цей метод досить важкий і доступний, лише досвідченим користувачам. При використанні даного методу не має можливості повернутися назад, доводиться проходити всі опції від початку до самого кінця.

Утиліта **menuconfig** запускається з командного рядка і являє собою псевдографічний інтерфейс з елементами меню. Елементи меню дозволяють задати потрібну конфігурацію. Якщо функція повинна бути включена в ядро, варто вибрати її і натиснути клавішу [Пробіл]. У порожніх дужках ліворуч від запису з'явиться зірочка (*). Для підключення модуля потрібно натиснути клавішу [**m**], і на екрані в дужках з'явиться символ **M**.

Інструментальний засіб **xconfig** запускається з середовища X Window, робота з ним ведеться через віконний інтерфейс із кнопками і меню. Активізація команд здійснюється за допомогою миші. Меню конфігурування являє собою набір кнопок. Щоб вибрати потрібну кнопку, досить клацнути на ній.

При використанні кожного з раніше описаних методів у каталозі **/usr/src/linux** створюється файл, якому привласнюється ім'я **.config**. У цьому файлі містяться обрані параметри, визначені на етапі конфігурування. Конфігураційний файл являє собою звичайний текстовий файл, який, в принципі, можна відредагувати вручну і, таким чином, внести в цей файл різноманітні зміни. Однак, використовувати подібний метод редагування файлу конфігурації не рекомендується.

Файл **.config** створюється після завершення етапу **make config**, **make menuconfig** чи **make xconfig**. Перед наступним конфігуруванням ядра доцільно зробити резервну копію цього файлу.

1.5 Компіляція вихідних кодів і інсталяція модулів (етап 4)

Для компіляції ядра варто виконати наступні кроки:

- **make dep**
- **make clean**
- **make bzlmage | bzdisk | bzlilo**
- **make modules**
- **make modules_install**

Перші з них — **make dep** і **make clean** — є підготовчими.

За допомогою команди **make dep** створюються файли залежностей (**.depend**) для уточнення того, яка частина вихідного коду компілюється на основі конфігурації користувача. Ці файли розташовуються в кожному з підкаталогів дерева вихідних кодів. Якщо немає серйозних порушень у розташуванні компонентів вихідного дерева, цей процес пройде спокійно — будуть лише створені файли **.depend**.

Команда **make clean** використовується для видалення допоміжних файлів від попередніх процесів компіляції, які можуть вплинути на хід поточного процесу формування.

За підготовчими командами слідують команди, що фактично збирають ядро. Користувачу доведеться вибрати одну з трьох команд: **make bzImage**, **make bzdisk** чи **make bzlilo**. Кожна з них виконує практично ту саму операцію; дві останні — виконують одну додаткову дію.

Команда **make bzlmage -** це стандартна операція, при якій відбувається тільки компілювання ядра (і нічого більше). Якщо усі компоненти ядра скомпільовані коректно, то наново створене ядро буде розташоване в каталозі **/usr/src/linux/arch/1386/boot**. У цьому випадку новому ядру привласнюється відповідне ім'я **bzlmage**. Диспетчер завантаження, яким є LILO (чи GRUB) повинен знайти нове ядро і завантажити його. Для цього досить скопіювати файл **bzlmage** у необхідний каталог і виконати команду **lilo** для перевстановлення диспетчера завантаження.

Команда **make bzdisk** - дозволяє виконати практично ту ж задачу, що була виконана за допомогою команди **bzlmage**. Після завершення компіляції відбувається автоматичне копіювання файлу наново зібраного ядра на гнучкий диск. Надалі цей диск може бути використаний для завантаження системи; нове ядро буде активним. Цей метод зазвичай використовується для перевірки ядра в системі, у якій воно було зібрано, чи ж для тестування ядра на інших апаратних засобах.

Команда **make bzlilo** – це рекомендований метод формування і інсталяції нового ядра, що вимагає попередньої підготовки **lilo**. При використанні цього методу зв'язаний map-файл ядра не переміщається в інший каталог. Більш того, зібране ядро може бути записане поверх вже існуючого, причому записано з помилками, тому його використання не рекомендується. Проте цей метод дозволяє заощадити час на процесі установки ядра. Цей метод дуже схожий на раніше описаний метод **bzImage** і відрізняється тільки наявністю додаткової операції, що виконується після завершення компіляції ядра (виконується команда **lilo**, у результаті чого відбувається перевстановлення диспетчера початкового завантаження і, відповідно, розпізнавання нового ядра).

Процес зборки ядра завершується виконанням двох останніх команд — **make modules** і **make modules_install**. При виконанні команди **make modules** відбувається зборка модулів, що відповідають ядру, створеному на попередньому етапі. Команда **make modules_install**, у свою чергу, переміщає створені модулі з вихідного дерева ядра в каталог **/lib/modules/<kemel-version>/kernel/<module-type>**. Як тип модуля **(<module-type>)** використовується ім'я категорії, до якої відносяться створені модулі, такі, як **block, fs, ipv4, ipv6, misc, net, pcmcia** і так далі.

Нижче представлена проста однорядкова команда, що дозволяє цілком виконати процес зборки ядра:

make dep; make clean;

make bzlmage && make modules && make_modules_install

Команди, відділені крапкою з комою, виконуються послідовно. Команда, перед ім'ям якої знаходяться символи &&, виконується тільки в тому випадку, якщо попередня команда буде завершена без помилок. Це дозволяє уникнути виконання **make modules** при одержанні помилок під час процесу формування ядра на етапі **make bzlmage**. Дана властивість може бути використана для виводу відповідних повідомлень про помилки. Крапка з комою і подвійні символи & працюють практично з будь-якою командою.

1.6 Переміщення файлів ядра (етап 5)

Після завершення процесу зборки необхідно встановити як ядро, так і його map-файл у каталог, де вони будуть постійно знаходитися. У більшості дистрибутивів розроблювачі відмовилися від практики розміщенні ядра в кореневому каталозі. В даний час ядро інсталюється, зазвичай, в каталог **/boot**, який створений безпосередньо для збереження ядра і його map-файлів.

Для цього потрібно скопіювати (чи перемістити) файл **System.map** та ядро у каталог **/boot**, додавши до них номер версії ядра:

cp System.map /boot/System.map-2.4.10

Потім точно так само скопіюйте (чи перемістіть) ядро:

cp arch/i386/boot/bzlmage /boot/bzImage-2.4.10

1.7 Конфігурування і запуск LILO (етап 6)

Диспетчер початкового завантаження **LILO** дає можливість визначити кілька образів завантаження. Вставивши новий розділ образа у файл **/etc/lilo.conf**, можна одержати додатковий образ завантаження. Нижче приведений приклад конфігурації **LILO**, що містить у собі єдиний образ завантаження:

```
boot = /dev/hda
install = /boot/boot.b
message = /boot/message
prompt
timeout = 50
default = linux
image = /boot/vmlinuz-pc97-2.2.14-modular
    label  = linux
    root   = /dev/hdal
    read-only
```

Для добавлення нового образу завантаження необхідно продублювати шість рядків розділу визначення завантажувального образу, потім модифікувати їх для другого образу, щоб зробити його доступним під час завантаження. Для коректного виконання завантаження доведеться змінити рядок **image**, а також значення, задані в рядку **label**. Дані, що знаходяться в рядку **image**, містять у собі ім'я ядра, що було скопійовано на попередньому етапі: **/boot/bzlmage-2.4.10**. Значення, дані в рядку **label**, повинні бути унікальні. Рядок **label**

являє собою унікальний ідентифікатор, що дозволяє вибрати один з образів завантаження, параметри якого встановлені у файлі **/etc/lilo.conf**. Цілком змінений файл **/etc/lilo.conf** виглядає приблизно так:

```
boot = /dev/hda
install = /boot/boot.b
message = /boot/message
prompt
timeout = 50
default = linux
image = /boot/vmlinuz-2.4.10
    label  = linux
    root   = /dev/hdal
read-only

image = /boot/vmlinuz-pc97-2.2.14-modular
    label  = linux.orig
    root   = /dev/hdal
    read-only
```

Після завершення конфігурації файлу **/etc/lilo.conf** варто переустановити диспетчер початкового завантаження. Це дозволить ввести новий образ у якості однієї з опцій завантаження. Для цього потрібно виконати наступну команду:

```
# lilo
Added linux *
Added linux.orig
```

Після завершення цього етапу потрібно перезавантажити систему. До кожної з обраних опцій можна звернутися за допомогою запрошення LILO. У тому випадку, якщо нове ядро є причиною некоректної роботи системи під час її завантаження, можна скористатися вихідним ядром.

2. Порядок виконання роботи:

2.1 Запустити систему та залогуватись.
2.2 Відкрити для редагування файл /etc/lilo.conf.
2.3 Провести короткий опис кожного пункту конфігурації завантажувальника.
2.4 Змінити назви міток пунктів завантаження та закрити файл зберігши зміни. Короткий опис проведених дій.
Інсталювати завантажувальник.
2.5 Перезавантажити систему та перевірити зміни в завантажувальнику. Короткий опис виводу на екран при завантаженні та висновки.
Завершити роботу системи.
2.6 Компіляція ядра ОС Linux.
Порядок виконання:
а) визначити версію ядра яке встановлене.
б) строрити загрузочну дискету.
в) витерти непотрібні файли при попередній компіляції.
г) провести конфігурацію ядра ОС.
д) встановлення залежностей між файлами конфігурації.
г) підготовка вихідних текстів для компіляції.
д) компіляція ядра.
е) компіляція модулів ядра ОС.
є) встановлення модулів ядра.
з) скопіювати нове ядро ОС у відповідні каталоги.
ж) провести конфігурацію завантажувальника.
й) завантажити систему з новим ядром.
Завершити роботу системи.

Необхідні команди:

uname –r	-версія ядра;
/sbin/mkbootdisk 2.4.20-8	-створює загрузочний диск;
make mrproper	-витирає непотрібні файли;
make menuconfig	-конфігурація ядра;
make dep	-встановленя залежностей;
make clean	-підготовка вихідних текстів;

make bzImage	-компіляція ядра;
make modules	-компіляція модулів;
make modules_install	-встановлення модулів ядра;
make install	-копіювання нового ядра в необхідні каталоги.

3. Порядок оформлення звіту по роботі.

3.4 На титульній сторінці вказати назву університету, кафедри, назву і номер роботи, прізвище, ініціали, номер групи виконавця, прізвище і ініціали викладача, який керував роботою, рік виконання роботи.
3.5 Вказати мету роботи.
3.6 Привести завдання для розробки.

4. Контрольні запитання.

1. Для чого призначений завантажувальник LILO?
2. З якого файла беруться параметри завантажувальника системи?
3. Що вказується в стрічці image?
4. Яка необхідна команда для інсталяції завантажувальника?
5. Яка команда для конфігурації ядра ОС?
6. Яка команда для встановлення залежностей між файлами конфігурації ядра?
7. Якою командою компілювати ядро?
8. Який файл відповідає за конвігурацію загрузки ОС Linux ?

Список літературних джерел

1. Матчел Марк. Программирование для Linux. Профессиональный подход / Матчел Марк, Оулдем Джеффри, Самьюэл Алекс: Вильямс, 2004. – 288с.

2. Матт Уэлш. Запускаем Linux / Матт Уэлш, Маттиас Калле Далхамер, Лар Кауфман: Символ-Плюс, 2000. – 832с.

3. Виктор Костромин. Самоучитель Linux для пользователя / Виктор Костромин : БХВ- Петербург, 2005. – 658с.

4. Роберт Лав. Разработка ядра Linux / Роберт Лав: Вильямс, 2008. – 448с.

5. Эви Немет. Руководство администратора Linux / Эви Немеет, Гарт Снайдер, Трент Хейн: Вильямс, 2005. – 880с.

6. Девид Тейнсли. Язык shell. Linux и Unix / Девид Тейнсли: BHV, 2001. – 512с.

7. Кай Петцке. Linux от понимания к применению / Кай Петцке: ДМК, 2000. – 576с.

8. П. Фолькердинг. Установка и конфигурирование Linux (+2CD-ROM) / П. Фолькердинг, К.Рейчард, Э.Фостер-Джонсон: Питер, 1999. – 496с.

Зміст