Міністерство освіти і науки України
Сумський державний університет

5466 Методичні вказівки
до виконання лабораторних робіт
із дисципліни «**Мережеві операційні системи**»
для студентів спеціальності
172 *«Телекомунікації та радіотехніка»*
денної форми навчання

Частина 2

Суми
Сумський державний університет
2022

Методичні вказівки до виконання лабораторних робіт із дисципліни «Мережеві операційні системи» / укладачі: В. В. Гриненко, О. В. Д’яченко. – Суми : Сумський державний університет, 2022. – 48 с.

Кафедра електроніки і комп’ютерної техніки

Лабораторна робота 4

Права доступу в ОС Linux

Мета – набуття знань щодо прав доступу в ОС Linux і практичних навичок щодо використання командних ресурсів за зміни прав доступу до файлів і директорій.

Завдання для самостійної підготовки

1. Вивчити:
   - поняття «право доступу» і «метод доступу»;
   - атрибути доступу до файлів в UNIX;
   - перегляд інформації про права доступу;
   - зміна прав доступу.
2. Детально ознайомитися з довідкової системи *man* із такими командами UNIX: *ls –l, chmod, chown, umask.*

Теоретичні відомості

1. Поняття прав доступу користувача

Будь-яка UNIX-подібна ОС є система багатокористувацька, у ній передбачено механізм, який обмежує доступ користувачів до файлів і директорій. Доступ означає не лише можливість читати чи змінювати вміст окремих файлів, а й можливість створювати файли (директорії), вилучати їх, запускати файли, якщо вони є виконувані, змінювати їхні назви, а також змінювати всі ті атрибути, що й визначають право доступу, тобто хто саме може чинити з цим файлом чи директорією.

Отже:

– кожен процес має ідентифікатор користувача (userID); зазвичай він збігається з userID того користувача, який запустив цей процес;

– процеси, запущені автоматично, теж мають userID, начебто їх запустив реальний користувач; чий саме userID отримують ці програми, зазвичай визначається тими програмами, які стартують;

– деякі програми в процесі виконання можуть змінювати свій userID і відповідно набувати прав, яких сам користувач не

мав. До речі, зміна userID здійснюється не лише коли потрібно розширити права програми, а й навпаки – обмежити їх до прав певного конкретного користувача;

– якщо процес може змінювати свій userID, то розрізняють реальний userID та ефективний. Реальний userID – це ідентифікатор користувача, який запустив процес. А ефективний – це новий userID, якого задача отримала під час виконання;

– права на файл (чи директорію) визначаються за ефективним userID процесом.

Усі користувачі для кожного файлу (чи директорії) поділяються на три категорії:

– власник цього файлу;

– група з особливими правами доступу до цього файлу;

– усі решта.

Це означає, що можна встановити три різних набори прав доступу для кожного файлу чи директорії. Один такий набір визначатиме права користувача, який є власником файлу; інший набір визначатиме права для користувачів, які належать до певної групи, але не є власниками, і, врешті, третій набір установлює права для всієї решти користувачів, які не належать до цієї групи з особливими правами доступу і не є власниками файлу. Отже, у кожного файлу (директорії) є три атрибути, які зберігаються десь у заголовку файлу й також регулюють доступ до нього. Звичайно атрибутів у файлі є не три, а більше. До атрибутів можна віднести ім'я файлу, його обсяг, час створення тощо. Але в цьому разі нас цікавлять лише ці три.

У заголовок файлу записується ідентифікатор користувача userID, який вважається його власником. Водночас власником може бути лише один певний користувач. Крім того, як атрибут записується ідентифікатор групи (groupID), який і визначає групу з особливими правами доступу, про яку йшлося вище. Кожна така група (її назва, числовий ідентифікатор – groupID і склад) визначається адміністратором системи. Тобто пересічний користувач, навіть якщо він і є власник файлу, не може довільно скласти список «близьких друзів», яким він довіряє особливі права щодо свого файлу. Він може лише обрати придатну групу

з наявних лише, якщо він сам входить до цієї групи. В атрибутах файлу є певний набір бітів чи «прапорців», який і зазначає – хто саме й що саме може чинити з цим файлом. Цей набір називається permissions, що можна перекласти як «права на доступ».

Під час набору команди *ls –l* на моніторі висвітлюється:

-rw-r--r--	*st1*	*wheel*	**4297**	**23мар 17:37**	*list_us*
-rwxr-xr-x	*st1*	*wheel*	**1502**	**17мар 12:03**	*myProg*
-rw-r--r--	*st1*	*wheel*	**5354**	**12мар 23:51**	**tmp.*dat***
права	власник	група	довжина	дата	ім'я файла

У другому стовпчику подано ім'я (login name) користувача – «власника» цих файлів (у цьому разі це – st1). У третьому стовпчику – назва групи, приписаної також до цих файлів (у цьому разі – wheel). У першому стовпчику набір знаків типу *r* (читання), *w* (запис) та – (нічого) зазначає можливості для всіх трьох категорій користувачів. Необхідно зауважити, що в самому заголовку файлу зберігаються не імена користувачів і груп, а їхні числові номери, а права насправді являють собою не ланцюжок літер, а набір бінарних бітів. Просто команда *ls* зображує їх у більш узвичаєному вигляді.

2. Основні біти доступу (читання/запис/виконання)

Під час роздрукування вмісту директорії (наприклад, командою *ls*) кожен рядок має вигляд

-rw-r--r-- root wheel 178 Mar 19 21:58 io.c

Він складається з десяти символів. Проте перший символ не має відношення до permissions, а позначає тип цього об'єкта. У директорії, окрім файлів, можуть міститися піддиректорії та інші. Тому перший символ і зазначає, що за об'єктом ми спостерігаємо: звичайний файл (позначка -), піддиректорія (позначка *d*) чи ще якийсь специфічний об'єкт (*l, s, p*...). Інші дев'ять знаків насправді являють собою три групи по три символи.

Кожна така група визначає права для якоїсь із трьох категорій користувачів:

– перша група – права власника;

– друга група – з особливими правами доступу;

– третя група – права для всієї решти.

Зміст окремих бітів у кожної такої групи прав є однаковий для всіх трьох категорій користувачів, тому можна докладніше розглядати кожну таку групу, не уточнюючи, для якої саме категорії користувачів вона призначена. Проте для файлів та директорій зміст цих бітів не набагато відрізняється, тому їх потрібно розглянути окремо.

Під час роботи з файлом перший біт позначається літерою *r* (read) і означає, що користувачеві, який підпадає під відповідну категорію, дозволяється читати вміст цього файлу. Користувач може переглянути вміст файлу, а також скопіювати цей файл, але це не означає, що якщо це є програма, то користувач зможе запустити його на виконання. Другий біт позначається літерою *w* (write) і дозволяє запис у файл. Користувач може змінити вміст файлу (наприклад редактором), дописати дещо наприкінці або стерти увесь вміст. Проте цей біт ще не надає права вилучити сам файл із директорії чи змінити його назву, тому що це визначається правами на саму директорію, але надає можливість зробити цей файл порожнім чи скопіювати в нього вміст іншого файлу й тим самим підмінити його. Третій біт позначається літерою *x* (execute), дозволяє запустити на виконання цей файл, якщо він являє собою програму чи командний файл.

Для директорій перший біт *r* дозволяє читати вміст цієї директорії, тобто список файлів і піддиректорій, які містяться в ній. Проте цей біт ще не надає можливості зайти в цю директорію (командою *cd*) чи дістати доступ до вмісту, тобто читати, запускати, змінювати файли, навіть якщо права доступу, установлені на самих файлах, це дозволяють. Тому само собою право читання директорії практично не є чинним і цей біт ставиться лише разом із бітами *x*. Для директорій біт – *x* означає, що користувач може дістати доступ до компонентів, тобто

окремих файлів і піддиректорій. Лише за наявності цього біта система дозволить увійти в цю директорію й виконати певну дію з файлом, якщо самі файли це дозволять.

Також, якщо навіть внутрішні піддиректорії мають нормальні права для певної категорії користувачів, а вища директорія – не має, бо відсутній біт *x*, то цим користувачам не вдасться зайти в піддиректорії, оминаючи вищу. Система перевіряє увесь шлях до кінцевої директорії файлу (наприклад, */usr/share/misc/fonts*), і, якщо хоча б один з компонентів цього шляху не має відповідного біта, користувачеві буде відмовлено в доступі. Зрештою біт *w*, встановлений на директорії, дозволяє змінювати вміст директорії, тобто дозволяє створювати нові файли (чи копіювати інші файли в цю директорію), змінювати назви файлів і вилучати файли.

Як уже зазначалося, якщо права на директорію не дозволяють користувачеві вилучити файл, який розміщено в ній, бо немає біта *w*, це ще не означає, що користувач не зможе вилучити вміст файлу, наприклад, текстовим редактором. З іншого боку, якщо користувач має право змінювати вміст директорії, він зможе вилучити чи перейменувати кожний файл, який вміщено в ній, навіть якщо права на самому файлі не дозволяють йому не те що писати у файл, але й читати його.

Треба також звернути увагу на те, що жодні з перелічених тут прав не мають відношення до змін самих атрибутів доступу, тобто – власника файлу, групи та прав доступу. Їхня зміна підпорядковується іншим законам, про які йтиметься нижче.

Також усі ці біти не мають жодного значення для користувача root, тобто він може робити з файлом чи директорією все що завгодно. Проте і тут є один виняток. Оскільки біт x на файлі є основною ознакою виконуваності цього файлу, навіть root не зможе переконати систему, що файл є програмою і його можна виконувати, поки не встановить в атрибутах цей біт.

3. Додаткові біти доступу

Крім розглянутих вище бітів, що встановлюються роздільно за трьома категоріями користувачів, є ще три біти доступу, які можна віднести до файлу в цілому, оскільки їхня дія не залежить від того, який користувач, з якої категорії намагається звернутися до файлу. Та й призначення цих бітів полягає не в обмеженні доступу до файлу чи директорії, а у зміненні певних властивостей файлів директорій.

Біт ***suid*** розшифровується як Set user ID, перекладається як «установити ідентифікатор користувача». Призначення його полягає в тому, що якщо його установлено на файлі, який є програмою, то під час виконання ця програма автоматично змінює ефективний userID на ідентифікатор того користувача, який є власником цього файлу. Тобто, незалежно від того, хто запускає цю програму, вона під час виконання має права власника цього файлу. Зазвичай це роблять для того, щоб користувач міг виконати дії, які потребують привілеїв root (наприклад, змінити свій пароль), а власником такої програми повинен бути користувач root.

Зрозуміло, що така програма є потенційно небезпечною. За нормальних обставин вона не дозволить звичайному користувачеві зробити те, що виходить за межі його повноважень, наприклад, програма passwd дозволить користувачеві змінити лише власний пароль, але не паролі інших користувачів. Але навіть незначна помилка в такій програмі може призвести до того, що зловмисник зможе змусити її виконати ще якісь дії, не передбачені автором програми.

Біт ***sgid***. Розшифровується як Set group ID, перекладається як «встановити ідентифікатор групи». Цей вміст є аналогічний вмістові попереднього біта, лише змінюється не ідентифікатор користувача, а ідентифікатор групи. Тобто під час виконання цього файлу він має такі права, начебто його запустив дехто із групи, приписаної до цього файлу.

Для FreeBSD (й інших BSD-систем) цей біт, знову таки, не чинить жодної дії. Але в деяких інших UNIX-системах він

означає, що коли файли створюються в такій директорії, в їхніх атрибутах проставляється та сама група, що й у директорії. Тобто файли, створювані в такій директорії, «успадковують» групу від директорії.

Біт ***sticky***. Для директорій його значення полягає в тому, що вилучити файл із такої директорії (чи перейменувати) здатен лише власник файлу. В звичайному випадку (без такого біта) можливість вилучати файли (як і створювати) визначається правом запису на директорії. Тобто, якщо якийсь користувач належить до категорії, для якої дозволено запис у директорію, він може вилучити в ній який завгодно файл, незалежно від атрибутів самого файлу.

4. Поєднання бітів доступу

Зазвичай права на файл (наприклад, для всієї решти) встановлюються в такий спосіб:

--- жодних прав (не можна ані читати, ані змінювати вміст);

r-- лише читання;

rw- і читання, і запис (зміна) файлу.

Якщо файл є виконуваний, то права можуть виглядати так:

--- жодних прав (читати не можна, запускати не можна);

r-x можна запустити файл-задачу на виконання;

rwx можна не лише запустити, а й дещо у ньому змінити.

Інші поєднання (наприклад, -*w*- чи --*x*) здаються безглуздими. Проте це не завжди так. Розглянемо докладніше.

Права *w* означають, що користувач із відповідної категорії не може прочитати цей файл, але може в нього писати.

Для виконуваних файлів права --*x* є насправді цілком законна комбінація. Вона означає, що запустити цю задачу можна (й отримати від неї певний результат). Комбінація -*wx,* швидше за все, значення не має, як і просто *w* на файлі, який виконується. Ви лише надаєте можливість змінити вміст. Результатом може бути лише псування вашого файлу. Також можна віднести до безглуздих і права *r*-- на файлі, що виконується. Інший користувач спокійно може скопіювати цей файл собі, водночас він стає повноправним власником

отриманої копії. Після цього він поставить на свою копію такі права, які йому заманеться, і, врешті-решт, запустить його.

Якщо відсутній біт доступу до каталогу --*x* (доступ до вмісту), то кожна комбінація з двох інших нічого корисного не надасть. Комбінація *r*-- надасть можливість отримати список файлів у цій директорії, наприклад, командою *ls* і не більш того. У таку директорію не можна увійти командою *cd*. І навіть якщо усередині її піддиректорії мають цілком нормальні права доступу (наприклад, *r-x*), потрапити в них буде неможливо.

Комбінації -*w*- і *rw*- мають ще менше сенсу. Змінити щось у директорії з такими правами доступу не вдасться.

А ось поєднання --*x* і -*wx* є насправді цілком осмислені. У такий спосіб ніхто сторонній не зможе переглянути, що за файли й директорії в ній розміщено. Але, якщо ви когось повідомите, як називається файл, що міститься там, його можна без проблем вилучити звідти чи переглянути безпосередньо на місці.

Права доступу до файлів задаються під час створення файлу і в подальшому можуть бути змінені командою

chmod <нові права> <файл(и)>

<нові права> задаються двома способами. Перший – символьний. Використовуються такі позначення: *u* – власник файлу (user), *g* – група (group), *o* – всі інші (others), *a* – всі категорії користувачів одночасно (all). Після категорії користувачів задається дія: ‘+’ – додати права до існуючих, ‘-’ – відібрати права (якщо існують), ‘=‘ – встановити права замість існуючих. Далі позначаються права: *r* – зчитувати (Read), *w* – записувати (Write), *x* – виконувати (eXecute). Можна формувати список зміни прав, розділяючи окремі категорії користувачів комами (без пробілів). Наприклад, команда

chmod u+rw,g=rx,o-w my_file

встановить для користувача права на зчитування та записування (право на виконання для користувача буде залежати від того, чи було воно встановлено раніше), для групи встановить права на зчитування та виконання (незалежно від того, які права були раніше), а для всіх інших гарантовано заборонить записування (права на зчитування та виконання

будуть встановлені залежно від того, чи були вони встановлені раніше).

Другий – числовий спосіб задавання прав доступу – зручніше використовувати, коли потрібно встановлювати нові права незалежно від попередньо встановлених. Водночас використовується подання у восьмеричній системі числення, яке легко зрозуміти з наведеної таблиці 4.1.

Таблиця 4.1 – Таблиця подання прав доступу

Восьмеричне подання	**Двійкове подання**	**Права доступу**
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Числова нотація команди *chmod*

Набір прав разбивається на 4 трійки:

sst rwx rwx rwx

і розглядається в вигляді бітового поля: біт встановлений, якщо є відповідне право. Кожна трійка бітів записується десятковим числом.

Наприклад, команда

chmod 0751 my_file

встановлює такі права доступу до файлу my_file: власнику – зчитування, записування, виконання, групі – зчитування та виконання, а всім іншим – лише виконання. Команда *ls -l* покаже для цього файлу таку інформацію про права доступу: *-rwxr-x--x* (перший «мінус» означає, що це звичайний файл).

За замовчуванням для всіх нових файлів, що створюються, встановлюються такі права: для файлів 666, тобто *rw-rw-rw-*, для каталогів – 777, тобто *rwxrwxrwx*.

Є особлива команда – *umask,* яка дозволяє зменшити права доступу, що встановлюються для нових файлів. Параметром цієї команди є восьмерична маска, аналогічна тій, що використовується за числового способу задавання параметрів команди *chmod*, але у разі команди *umask* права доступу, що задані нею, будуть відніматися від прав доступу, що були задані по замовчуванню для вже утвореного файлу. Команда *umask* не може додавати прав, тому нові файли ніколи автоматично не отримають право на виконання, в разі необхідності його потрібно буде додати вручну. Дія команди *umask* поширюється на всі файли всіх типів, що утворюються впродовж поточної сесії користувача після цієї команди.

Наприклад, після команди *umask 123* всі файли будуть утворюватись із параметрами доступу *rw-r--r--* (644), а всі каталоги – *rw-r-xr--* (654). За замовчуванням рекомендується використовувати команду *umask 22* (інтерпретується як *umask 022*).

Для зміни власника файлу існує команда *chown*. З міркувань безпеки, ця команда дозволяє встановлювати власником файлу будь-кого лише системному адміністратору (root'у). Інші користувачі можуть лише привласнити файл собі, якщо вони мають на це права, встановлені правами доступу до файлу та каталогу.

Приклади:

1) додати групі право на запис:
chmod g+w file;

2) прибрати в інших права на запис та виконання:
chmod o-wx file;

3) додати власнику та групі права на читання й запис:
chmod ug+rw file;

4) встановити права інших такими самими, як у групі
chmod o=g file;

5) встановити права інших і групи такими самими, як у власника

chmod og=u file;

6) декілька змін можна перераховувати через кому.

Додати власнику право на виконання, а у групі та інших прибрати право на запис:

chmod u+x,go-w file.

Порядок виконання роботи

Завдання 1

1. Створіть нового користувача tester1. Задайте йому пароль. Прослідкуйте зміни у файлах */etc/passwd*, */etc/group*, */etc/shadow* до початку створення користувача та після виконання кожної з команд.
2. Створіть нового користувача tester2 з домашнім каталогом */home/new_tester*, UserID=1020, GroupID=1050. Додайте користувача tester1 в базову групу користувача tester2.
3. Не змінюючи пароля користувачу tester2 забороніть йому доступ у систему, залишивши доступ у систему користувачу tester1.
4. Забороніть доступ у систему всім непривілейованим користуачам.

Завдання 2

5. Створити два каталоги priv_*5 букв прізвища* та publ_*номер групи_номер за списком.*
6. Створити 2 файли в щойно створених каталогах file_*номер за списком_priv* та file_*номер групи*_pub.
7. Переглянути дозволи на цих файлах. Пояснити одержані результати.
8. Переглянути дозволи на всі файли та підкаталоги в поточному каталозі.
9. Видалити дозволи інших для читання та виконання для каталогу priv_
10. Перевірити зміну дозволів.

11. Навести 2 приклади символьного методу зміни дозволів.
12. Додати дозвіл на запис для інших для каталогу pub_
13. Видалити будь-який дозвіл від групи та інших користувачів файлу _priv.
14. Надати всім користувачам однаковий дозвіл на читання та запис для файлу _pub.
15. Створити test.sh файл, що містить вміст " date" в каталозі */tmp*.
16. Виконати test.sh файл; Описати результат.
17. Створити файл, для якого будуть встановлені права: *rw* – для власника та інших для всіх *rwx.*
18. Прибрати право на запис і читання для інших.
19. Дозволити доступ до свого домашнього каталогу.
20. Перейти в каталог сусіда та створити там файл
21. Створити вкладені в каталоги dir 1→ dir 11→dir 111.
22. Дати можливість усім записати файл в dir 111.
23. Створити файл у каталозі dir 111 сусіда.
24. Зробити свій каталог недоступним.
25. Створити новий каталог і файл. Подивитися, які права система їм назначила, використовуючи команди (*touch f; ls -l f; mkdir D; ls -l D*).

Завдання 3

1. Створити новий каталог і файл. Подивитися, які права система їм назначила, використовуючи команди (*touch f; ls -l f; mkdir D; ls -l D*).

2. За встановленою маскою та значенням прав, отриманих під час створення файлів і каталогів, визначити початково встановлені в системі права.

3. Задати маску, що дає можливість створювати файли та директорії, доступні лише власникові та іншім. Перевірити дію маски.

Завдання 4

1. Встановити первинне значення маски. Створити структуру директорій із лабораторної роботи 1 згідно зі своїм варіантом.

2. Зробити одну з директорій 3-го рівня загальнодоступною з читання та запису.

3. На батьківську директорію для директорії з п. 4 поставити такі права, щоб лише власник директорії міг створювати файли в директорії з п. 4. Перевірити дію прав, створюючи файли в директоріях п. 4 і п. 5 з-під різних користувачів.

4. Встановити на одну з директорій 1-го рівня такі права, що б усі могли лише створювати та видаляти в ній файли і директорії, але не могли проглянути вміст.

5. В одній з директорій 1-го рівня створити два файли file1 і file2. За допомогою числових значень прав зробити file1 загальнодоступним для читання та запису, а file2 доступним всім лише для читання.

6. На одну з директорій 2-го рівня поставити такі права, щоб усі могли створювати в ній файли та директорії, але видаляти файли та директорії могли лише їх власники.

7. У директорії з п. 8 створити файли file3 і file4. Встановити на них такі права: file3: *-rwxr-xr-x* file4: *-rwxr-x---*.

8. Накреслити схему створених директорій і файлів. В об'єктів файлової системи, права яких мінялися порівняно з первинними, на схемі позначити права.

Завдання 5

1. Створіть каталог lab_3.

2. Скопіюйте в каталог lab_3 файл */bin/cat* під назвою my_cat.

3. За допомогою файлу my_cat, що міститься в каталозі lab_3, перегляньте вміст файлу *.profile* (ви перебуваєте в домашньому каталозі).

4. Перегляньте список файлів у каталозі lab_3. Потім перегляньте список усіх файлів, враховуючи приховані, з

повною інформацією про файли. Зверніть увагу на права доступу, власника, дату модифікації файлу, що ви щойно скопіювали. Потім перегляньте цю інформацію про оригінальний файл (той, який копіювали) та порівняйте два результати.

5. Змініть права доступу до файлу my_cat так, щоб власник міг лише читати цей файл.

6. Переконайтеся в тому, що ви зробили ці зміни та повторіть п. 3.

7. Визначте права на файл my_cat так, щоб ви могли робити з файлом усе, що завгодно, а всі інші – нічого не могли робити.

8. Поверніться в домашній каталог. Змініть права доступу до каталогу lab_3 так, щоб ви могли його лише читати.

9. Спробуйте переглянути простий список файлів у цьому каталозі. Спробуйте переглянути список файлів з повною інформацією про них. Спробуйте запустити і видалити файл my_cat із цього каталогу.

10. За допомогою команди *su <user name>* завантажтесь у систему, користуючись обліковим записом іншого користувача (вам потрібно знати пароль цього користувача.) Спробуйте отримати доступ до Вашого каталогу lab_3. Перевірте, чи правильно зроблено завдання попереднього пункту. Створіть каталог lab_3_2.

11. Знову завантажтесь у систему, користуючись своїм обліковим записом. Спробуйте зробити власником каталогу lab_3 іншого користувача. Спробуйте зробити себе власником каталогу lab_3_2.

12. Зайдіть у каталог lab_3. Зробіть так, щоб нові створені файли та каталоги діставали права доступу згідно таблиці 4.2. Створіть новий файл і каталог і переконайтеся в правильності ваших установок.

Таблиця 4.2 – Таблиця варіантів до завдання

Варіант	**Права для файлів**	**Права для каталогів**
1	644	754
2	664	774
3	6-4	7-5
4	62-	73-
5	644	745
6	664	764
7	6-4	715
8	62-	63-
9	644	744
10	664	765

13. Поверніть собі права читати, писати та переглядати зміст каталогів.

Зміст звіту

1. Початкові дані та постановка задачі.
2. Текст програми.
3. Висновки.

Лабораторна робота 5

Текстовий редактор Vi

Мета – оволодіння практичними навичками роботи з редактором vi.

Завдання для самостійної підготовки

1. Вивчити:
   - завантаження редактора та вихід із нього;
   - завантаження та збереження файлів;
   - редагування тексту;
   - виконання команд UNIX.
2. Відповідно до завдання підготувати послідовність команд для його виконання.

Теоретичні відомості

Текстові редактори наявні в усіх операційних системах. Вони використовуються для створення та редагування текстових файлів. Текстові файли можуть містити документи, програмні коди та конфігураційні параметри. Документи можуть містити ознаки форматування (стиль і розмір шрифту, поля, відступи, тощо). Іноді такі ознаки можна розмістити безпосередньо в текстовому документі, використовуючи деякий визначений формат розмітки. Прикладами є формати TEX, а також сучасні HTML і XML.

Для потреб операційної системи більше значення мають класичні текстові редактори, важливою ознакою яких є те, що під час редагування вони відображають увесь вміст файлу, не приховуючи спеціальні символи чи ключові слова, а під час збереження файлу також не додають від себе додаткових керуючих символів. Це необхідно для можливості роботи з файлами, що містять коди програм або перелік конфігураційних параметрів. У системі UNIX текстовий файл може розглядатися системою як послідовність команд (так званий пакетний файл, або файл сценарію). Крім того, налаштовування практично всіх програм як тих, що входять до складу операційної системи, так і

прикладних програм здійснюється за допомогою текстових файлів.

Одним із засобів редагування текстових файлів є редактор vi. Він традиційно входить до будь-якої системи UNIX, і майже напевно наявний у кожній системі (звичайно адміністратор може його видалити, якщо віддає перевагу іншому редактору, але для зручності інших користувачів він не повинен так діяти). Редактор vi є єдиним текстовим редактором, який ви можете використовувати для редагування системних файлів без необхідності зміни прав доступу до цих файлів.

Редактор vi пропонує могутній набір операцій для редагування тексту, заснований на визначеній множині мнемонічних команд. Більшість команд викликаються натисканням одиночних клавіш і виконують прості функції редагування. На відміну від більшості сучасних редакторів vi взагалі не містить системи меню. Не має він і вбудованої системи підказок. Для роботи з цим редактором необхідно запам'ятати команди. Проте для тих, хто знає команди, ефективність роботи в цьому редакторі неперевершена.

Редактор vi відкриває «вікно» розміром з екран дисплея, в якому ви можете редагувати ваш файл. Під час редагування забезпечується зворотний візуальний зв'язок (ім'я vi – скорочення від слова "visual"). Нижній рядок екрану використовується як інформаційний і командний рядок.

Запуск редактора

vi [option...] [command...] [filename...]

view [option...] [command...] [filename...]

Команда view аналогічна vi, за винятком того, що автоматично встановлюється опція «лише зчитування» (-R). Під час використання view файл змінюватися не може.

Крім вже згаданої опції *–R*, у командному рядку vi допустимі інші опції, з яких потрібно відзначити *-r* (не плутати великі та маленькі літери!). Ця опція використовується під час відновлення, коли було ушкодження редактора чи всієї системи. *vi –r <filename>* відшукує останню збережену версію

зазначеного файлу. Якщо файл не визначений, то ця опція виводить список збережених файлів.

Редактор vi не виконує ніяких операцій редагування безпосередньо над зазначеним вами файлом. Замість цього він працює з копією вашого файлу, що є в буфері редагування. Коли ви активізуєте vi з одним аргументом – ім'ям файлу, цей файл копіюється в тимчасовий буфер редагування. Редактор запам'ятовує ім'я файлу, визначеного під час виклику, так, що пізніше він може скопіювати вміст буфера редагування назад у зазначений файл. Вміст заданого файлу не змінюється доти, поки всі зміни не будуть скопійовані назад у первісний файл.

Приклади команд, що можна застосовувати для входу в редактор vi:

vi	Редагує порожній буфер редагування.
vi <file>	Відкриває для редагування зазначений файл.
vi +123 <file>	Відкриває для редагування зазначений файл і переходить у ньому на рядок із номером 123.
vi +/<word> <file>	Відкриває для редагування зазначений файл і шукає перше входження слова "word".

Створення файлу

Щоб створити файл, введіть:

vi filename <CR>.

Коли ви введете команду vi з іменем файлу, vi очистить екран і відобразить вікно, в яке ви можете вводити та редагувати текст.

Режими роботи

У vi існує такі режими роботи:

Командний режим: у цьому режимі сигнал із клавіатури інтерпретується як команда редагування.

Режим вставки: перейти в цей режим можна набором будь-яких команд вставки, приєднання, відкриття, підстановки, зміни

чи заміщення, що існують в vi. У цьому режимі символи, набрані на клавіатурі, вставляються в буфер редагування.

Таблиця 5.1 – Команди перемикання між режимами

ESC	Переводить із режиму вставки в командний режим. Якщо не впевнені, в якому режимі ви є – краще зайвий раз натиснути ESC
	Команди вставки: переводять із командного режиму в режим вставки. Під час застосування команд вставки текст не знищується
I i	Команди ‘i’ та ‘I’ – “insert”. Команда ‘i’: починає вставляти текст перед символом, що розміщений під курсором. Для вставки нового рядка натисніть RETURN. Команда ‘I’ починає вставляти текст із початку поточного рядка
A a	Команди ‘a’ та ‘A’ – “append”. Команда ‘a’ працює точно так, як команда ‘i’, лише вставка тексту починається після курсору, а не перед ним. Це є однією з можливостей додавання тексту наприкінці рядка. Команда ‘A’ починає вставку тексту наприкінці поточного рядка
O o	Команди ‘o’ та ‘O’ – “open”. Відкривають новий рядок і вставляють текст. Команда ‘o’ відкриває новий рядок під поточним рядком, команда ‘O’ відкриває новий рядок над поточним рядком. Після того, як новий рядок відкритий, обидві команди працюють аналогічно команді ‘I’

Команди виходу

Існує кілька шляхів виходу з редактора vi. Якщо ви перебуваєте в режимі вставки, то перший крок – перейти в командний режим (клавіша **ESC**). Далі можна використовувати такі команди:

ZZ	Здійснює вихід із командного режиму vi. Вміст буфера редагування записується у файл, що редагувався, лише за умови, що були зроблені які-небудь зміни. Водночас ви повертаєтеся в ту командну оболонку, з якої був запущений редактор vi
ZQ	Здійснює вихід із командного режиму vi. Вміст буфера редагування не записується у файл, що редагувався. Водночас ви повертаєтесь в ту командну оболонку, з якої був запущений редактор vi.
:x	Діє так само, як і ZZ. Фактично означає перехід у режим ‘ex’ із наступним виконанням команди x
:q	Намагається здійснити вихід із редактора vi без запису з буфера редагування у файл. Проте, якщо з моменту останньої команди ‘w’ до буферу редагування вносилися зміни, то vi видає попереджувальне повідомлення, і вихід із редактора не здійснюється. Редактор vi також видає діагностику, якщо в списку аргументів залишилися імена ще невідредагованих файлів. Зазвичай ви хочете зберегти усі ваші зміни; для цього потрібно набрати команду ‘:w’
:q!	Діє так само як і ZQ. На відміну від попередньої, ця команда скасовує сеанс редагування. Знак оклику показує редакторові vi на необхідність безумовного виходу. Вміст буфера редагування не зберігається
:wq	Переходить у режим ‘ex’, зберігає зміни буфера редагування в поточний файл (команда w), після чого здійснює вихід із редактора (команда q)
:wq!	Скасовує перевірку, що звичайно виконується перед командою ‘w’. Наприклад, якщо ви володієте файлом, але не маєте дозволу на запис у нього, команда ‘:wq!’ дозволить вам змінити вміст

У режимі вставки можуть використовуватися такі символи:

BKSP	У поточному рядку повертає курсор назад на один символ
Ctrl+V	Відміняє спеціальне значення наступного набраного символу. Використовуйте **Ctrl+V** для вставки керуючих символів
Клавіші курсору	Можуть діяти нормально (переміщати курсор) або ні (вводити якісь незрозумілі символи) залежно від версії редактора та налаштовувань терміналу

Переважна більшість команд, що діють у командному режимі vi, в режимі вставки діяти не можуть, оскільки викликають вставку в буфер редагування набраних літер.

Команди редактора vi , що діють у командному режимі

Далі описані лише основні, найнеобхідніші команди. Більшість команд можуть використовувати лічильник, що стоїть перед ними та показує кількість повторів команди. Цей параметр у наступних описах команд не заданий, але він виконує свої функції, якщо не скасований яким-небудь аргументом, що стоїть перед ним. Коли редактор vi одержує команду неправильного формату, він сигналізує про це.

Переміщення курсору

Команди керування курсором (табл.. 5.2) дозволяють вам переміщати курсор по файлу. Вони дуже важливі тому, що деякі команди з інших груп (див. далі) використовують команди переміщення курсору як аргумент, що дозволяє визначити блок тексту. Поточна позиція курсору розглядається як початок блоку, а позиція курсору, яку він займе в разі виконання команди переміщення, – як кінець блоку (або навпаки, якщо переміщення здійснюється не вперед, а назад). Виконання таких команд відбувається лише після введення команди переміщення курсору.

Таблиця 5.2 – Команди переміщення курсору

l **SPACEBAR** →	Переміщає курсор на один символ уперед. Якщо заданий лічильник, то переміщає вперед на зазначену кількість символів. Ви не можете переміститися за межу кінця рядка
h **BKSP** ←	Переміщає курсор на один символ назад. Якщо заданий лічильник, то переміщає назад на зазначену кількість символів. Ви не можете переміститися за межу початку рядка
+	Переміщає курсор униз на початок наступного рядка
J **Ctrl-N** ↓	Переміщає курсор униз на один рядок, залишаючи його в тому самому стовпчику
K **Ctrl-P** ↑	Переміщає курсор на один рядок угору, залишаючи його в тому самому стовпчику. Якщо заданий лічильник, то курсор переміщається на зазначену кількість рядків
0	Переміщає курсор на перший символ поточного рядка
^	Переміщає курсор на перший символ рядка, такий, що не є міткою табуляції чи пробілом. Це дуже зручно під час редагування тексту з форматованими відступами (наприклад, тексту програми)
$	Переміщає курсор у кінець поточного рядка. Зверніть увагу, що курсор розміщений над останнім символом у рядку
w **b** **e**	Переміщає курсор уперед на початок наступного слова (w), назад на початок поточного слова (b), або на останній символ поточного слова (e), де слово визначене як рядок буквено-числових символів, розділених пунктуацією, мітками табуляції, символами нового рядка чи пробілами
%	Переміщає курсор на узгоджувальний роздільник,

	яким можуть бути кругла, операторна чи фігурна дужки. Це дуже зручно під час узгодження пар вкладених круглих, операторних і фігурних дужок (наприклад, під час редагування тексту програми)
(**)**	Переміщає курсор на початок речення. Переміщає курсор на кінець речення
{ **}**	Переміщає курсор на початок абзацу. Переміщає курсор на кінець абзацу
1G **G**	Переміщає курсор на початок файлу. Переміщає курсор на кінець файлу

Команди екрана

Команди екрана не є командами керування переміщенням курсору, і не можуть використовуватися як роздільники об'єктів тексту. Проте команди екрана здійснюють переміщення тексту та дуже зручні для сторінкової організації чи «прокручування» інформації з файлу на екрані дисплея.

Ctrl-U **Ctrl-D**	«Прогортає» екран на половину вікна вгору (Ctrl-U) чи вниз (Ctrl-D)
Ctrl-F **Ctrl-B**	Посторінково перегортає екран уперед і назад. Якщо можливо, то між сторінками зберігаються два нерозривні рядки. Заданий попереду лічильник зазначає кількість сторінок, на яку потрібно пересунути вперед чи назад
Ctrl-G	Показує стан редактора vi в інформаційному рядку: ім'я файлу, що редагується, чи був він змінений, номер поточного рядка, кількість рядків у файлі та відсоток файлу (в рядках), що передує місцеві перебування курсору
Ctrl-R **Ctrl-L**	Перемальовує вміст екрана. Використовуйте цю команду для витирання будь-яких системних повідомлень, що можуть накладатися на інформацію, що міститься на вашому екрані. Зверніть увагу, що системні повідомлення не впливають на файл, що редагується вами

Видалення тексту

Найбільш універсальна команда видалення тексту використовує як оператор виконання клавіші ‘d’. Цей оператор видаляє текстові об’єкти, обмежені поточною позицією курсору та командою переміщення курсору. Видалення фактично відбувається після завершення введення команди переміщення курсору. Вилучений текст завжди продовжує зберігатися в буфері.

BACKSPACE	Видалити поточний символ
1w	Видалити поточний слово
@	Видалити поточний рядок нового тексту або видалити увесь новий текст у поточному рядку. Командний режим
u	Скасувати останню команду
U	Відновити поточний рядок у колишньому стані
x	Видалити поточний символ
ndx	Видалити n-й об’єкт тексту x
dw	Видалити слово над курсором і наступний за ним пробіл або знак пунктуації
dW	Видалити слово та пунктуацію над курсором із наступною за ним прогалиною
dd	Видалити поточний рядок
D	Видалити частину рядка праворуч від курсора
d)	Видалити поточну пропозицію від поточної позиції курсора до кінця
d}	Видалити поточний параграф від поточної позиції до кінця

Переміщення тексту

Переміщення тексту здійснюється командами видалення, копіювання та вставки текстових блоків. Для цього зарезервовано 9 буферів видалення (позначених цифрами від *1* до *9*), 26 буферів для копіювання (позначених літерами від *a* до *z*) і один «безіменний» буфер, який використовується за

замовчуванням. Якщо необхідно зазначити певний буфер перед будь-якою командою копіювання або вставки вводяться подвійні лапки (*"*), а за ними – позначення буфера. Для команд вставки (*put*) позначення буфера може бути цифрою від *1* до *9* або літерою від *a* до *z*. Для команд копіювання (*yank*) позначення буфера може бути літерою від *a* до *z* або від *A* до *Z*. Якщо ви копіюєте текст у буфер з ім'ям '*A*' замість '*a*', то текст додається до вмісту буфера '*a*' (так само для будь-якого іншого буфера). Під час видалення вилучений текст автоматично заноситься в стек буферів, пронумерованих від *1* до *9*, тобто щойно видалений блок поміщається в буфер *1*, блок із буфера *1* переміщається у буфер *2* і так далі. Щойно вилучений текст також розміщається в «безіменному» буфері.

Отже, в буферах *1–9* розміщені блоки тексту, які було вилучено, в буферах *a–z* – блоки тексту, які було скопійовано, а в безіменному буфері – блок тексту, який було вилучено під час останньої операції або скопійовано без зазначення буфера.

Наприклад, команда *"4p* поміщає вміст буфера видалення з номером *4* у ваш буфер редагування під поточним рядком. Якщо ім'я буфера-джерела не зазначено, то текст вставляється з «безіменного» буфера. Отже, дуже просто видалити текст, потім пересунути курсор у те місце, куди ви хочете вставити вилучений текст, і після цього вставити текст у нове місце за допомогою команди '*p*' або '*P*'.

Пойменовані буфери найбільш зручні для збереження сукупності декількох частин тексту, які ви хочете мати постійно напоготові для пізнішого доступу до цих текстів, або їхнього переміщення і перерозміщення. Наприклад, для копіювання рядка з файлу в буфер '*a*' наберіть *"ayy*. Для того щоб помістити цей текст назад у файл, наберіть *"ap*.

Необхідно зазначити, що вміст пойменованих буферів не руйнується під час перемикання файлів. Тому ви можете видалити чи скопіювати текст у буфер, відкрити новий файл і потім виконати команду '*p*'. Вміст буферів губиться під час виходу з редактора, тому будьте обережні.

["<buf>]p	Вставляє текст із буфера <buf> у буфер редагування під поточним рядком чи після курсору залежно від того, містить буфер повний рядок чи ні
["<buf>]P	Вставляє текст із буфера <buf> у буфер редагування або над поточним рядком, або перед курсором залежно від того, містить буфер повний рядок чи ні
["<buf>]y <mov>	Копіює текст із буфера редагування в буфер <buf>. Аргумент <mov> обмежує об'єкт, що копіюється (аналогічно команді видалення 'd' – див. вище
["<buf>]yy ["<buf>]Y	Копіюють один рядок чи, якщо заданий лічильник, кілька рядків

Відміна/повтор операцій

u	Скасовує результат останньої команди вставки чи видалення. Після виконання команди вставки команда 'u' видаляє вставлений текст, а після команди видалення – вставляє текст назад
U	Відновлює поточний рядок у його первісному стані, перш ніж він був відредагований незалежно від того, скільки операцій редагування ви здійснили з того моменту, як ви перейшли на нього
.	Повторює останню команду вставки чи видалення

Пошук

Команди пошуку здійснюють пошук тексту, що відповідає заданому регулярному виразу та розміщений у буфері редагування, в прямому чи зворотному напрямку.

/[<pat>]┘ *?[<pat>]┘*	Здійснює пошук тексту, що відповідає шаблону <pat>, у прямому (/) чи зворотному (?) напрямку. Рядок є дійсним регулярним виразом. Якщо шаблон <pat> не заданий, то для пошуку використовується останнє значення <pat>, що використовувалось раніше (┘ – натискання клавіші “RETURN”)
n *N*	Повторює останню команду пошуку. Команда ‘*n*’ повторює пошук у тому самому напрямку, команда ‘*N*’ – у зворотному напрямку
f <ch> *F <ch>*	Знаходить символ <ch> у поточному рядку. Команда ‘f’ виконує пошук у прямому напрямку, команда ‘F’ – у зворотному
t <ch> *T <ch>*	Встановлює курсор над символом <ch>
; *,*	Знак “;” повторює пошук останнього символу. Знак “,” змінює напрямок пошуку на протилежний

Команди запису

Команди запису дозволяють вам переписувати увесь ваш буфер редагування або його частину в поточний або будь-який інший файл.

w [<file>]	Записує зроблені зміни у файл <file>, показуючи кількість записаних рядків і символів. Якщо параметр <file> не заданий, буфер записується в поточний файл. Якщо ім’я <file> визначене, то буфер редагування записується в цей файл. Редактор здійснює запис у файл лише, якщо це поточний файл і він редагується, або якщо файл не існує (в останньому випадку файл створюється). В інших випадках для запису ви повинні задати команду у зміненій формі – ‘w!’
w! <file>	Скасовує перевірку звичайної команди write і здійснює запис у будь-який файл, що дозволений із боку системи
w >> <file>	Додає вміст буфера до кінця

Команди зміни поточного файлу редагування

Для редагування файлу, іншого від того, що редагується в поточний момент, ви можете використовувати один із варіантів команди ‘e’.

e <file>	Використовується для початку редагування нового файлу. Редактор спочатку перевіряє, чи був модифікований буфер із моменту використання останньої команди ‘w’. Якщо це було зроблено, то видається попередження, і команда переривається. Якщо ні – команда видаляє вміст буфера редагування, робить файл <file> поточним і висвітлює нове ім’я файлу. Після перевірки, що цей файл дійсний (тобто не є бінарним файлом, каталогом або пристроєм), редактор читає файл у свій буфер. Якщо читання файлу виконане без помилок, у рядку стану з’являється кількість прочитаних рядків і символів. Поточним рядком спочатку вважається перший рядок файлу
e! <file>	Такий виклик скасовує повідомлення про модифікації, що були зроблені і не записані з буфера редагування, викликаючи тим самим скасування всіх змін, що були виконані перед редагуванням нового файлу
e +n <file>	Змушує редактор почати редагування не з першого, а з n-го рядка. Аргумент n може бути також командою редактора, що не містить пробілів, наприклад, +/pattern

Команди читання

Команди читання дозволяють вам читати текст у ваш буфер редагування з будь-якого місця (тобто, дозволяє вставити певний текст у визначене місце). Текст, що ви читаєте, повинен складатися, принаймні, з одного рядка, чи бути або файлом, або вхідною інформацією команди.

r [<file>]	Поміщає копію тексту з заданого файлу <file> в буфер редагування після поточного рядка. Якщо файл не заданий, то використовується поточне ім'я файлу. Якщо буфер редагування порожній, то це трактується як команда 'e'. Коли команда 'r' завершується успішно, то видається статистика, подібна до тієї, яка супроводжує виконання команди 'e'. Після команди 'r' поточним вважається останній прочитаний рядок
r! <cmd>	Читає вихідну інформацію команди <cmd> у буфер після визначеного рядка

Команди перемикання в shell

Часто виникає необхідність тимчасово вийти з редактора і виконати звичайні команди системи UNIX. Також може виникнути необхідність змінити робочий каталог, щоб редагування не впливало на цілісність іншого робочого каталогу. Нижче описані необхідні для цього операції.

cd <dir>	Зазначений каталог <dir> стає вашим поточним каталогом. Якщо каталог не визначений, то як ім'я цільового каталогу використовується домашній каталог користувача
sh	Запускається новий shell, в якому ви можете виконати будь-яку кількість команд. Для повернення в vi натисніть **Ctrl-D**
!<command>	Залишок рядка після знаку "!" передається в shell як команда для виконання. В текст команди <command> замість символів '%' і '#' підставляються імена поточного файлу й останнього файлу, що редагувався, а символ '!' заміняється на текст попередньої команди. Ці підстановки повторюються на екрані, але запам'ятовується рядок цієї команди без підстановок

Якщо з моменту останньої зміни в буфері редагування запис файлу не здійснювався, то перед виконанням команди видається попередження. Коли команда виконається, висвітлюється знак “!”.

Порядок виконання роботи

Необхідно розглянути такі питання роботи з редактором vi:

– режими роботи редактора, перемикання між ними;

– прості операції редагування: видалення символу, рядка, більше одного символу / рядків, розбивання рядків на два і об’єднання двох рядків у один;

– збереження тексту у файл із заданим або поточним ім’ям;

– завершення роботи з редактором: без збереження змін, зі збереженням зміненого тексту в файлі;

– перехід до заданого рядка;

– операції з буфером: копіювання/вирізання тексту в буфер, вставка тексту з буфера перед курсором/після курсору;

– вставка (читання) тексту з заданого файлу;

– операція скасування змін (undo);

– виконання команди оболонки без виходу з редактора;

– виконайте завдання відповідно призначеним варіантам.

Варіант 1

1. Завантажтеся в систему під вашим користувацьким ім’ям.
2. Створіть новий текстовий файл text за допомогою редактора vi. Наберіть наведений нижче текст:

This is an example.
Test1, Test2, Test3
alpha / beta / gamma.

3. Збережіть набраний текст у файл test1.
4. Виконайте команду *ls*, не виходячи з редактора, переконайтеся, що файл test1 створений.
5. Поставте курсор перед словом Test2. Видаліть 3 символи.

6. Перемістіть 3-й рядок перед першим (вирізання / вставлення).

7. Вставте текст з файлу testl перед другим рядком.

8. Перейдіть у початок 1-го рядка.

9. Видаліть 2 рядки однією командою.

10. Завершіть роботу з редактором без збереження зроблених змін.

Варіант 2

1. Завантажтеся в систему під вашим користувацьким ім'ям.

2. Створіть новий текстовий файл text за допомогою редактора vi. Наберіть наведений нижче текст:

This is an example.

Test1, Test2, Test3

alpha / beta / gamma.

3. Збережіть набраний текст у файл textfile.

4. Виконайте команду ***ls***, не виходячи з редактора, переконайтеся, що файл textfile створений.

5. Поставте курсор на початок 1-го рядка. Видаліть 8 символів.

6. Потрібно скопіювати 2-й рядок і вставити його двічі після останнього рядка.

7. Вставте вміст файлу textfile в початок тексту.

8. Перейдіть на початок 1-го рядка.

9. Видаліть 4 рядки однією командою.

10. Завершіть роботу з редактором зі збереженням зроблених змін.

Варіант 3

1. Завантажтеся в систему під вашим користувацьким ім'ям.

2. Створіть новий текстовий файл text за допомогою редактора vi. Наберіть наведений нижче текст:

This is an example.

Test1, Test2, Test3

alpha / beta / gamma.

3. Збережіть набраний текст у файл file.

4. Виконайте команду ***ls***, не виходячи з редактора, переконайтеся, що файл file створений.

5. Поставте курсор на початок 2-го рядка. Видаліть 2 символи.

6. Скопіюйте 3-й рядок і вставте його перед першим рядком.

7. Вставте вміст файлу file на кінець тексту.

8. Перейдіть на початок 3-го рядка.

9. Видаліть 2 рядки однією командою.

10. Завершіть роботу з редактором зі збереженням зроблених змін у файлі file2.

Варіант 4

1. Завантажтеся в систему під вашим користувацьким ім'ям.

2. Створіть новий текстовий файл text за допомогою редактора vi. Наберіть наведений нижче текст:

This is an example.

Test1, Test2, Test3

alpha / beta / gamma.

3. Збережіть набраний текст у файлі testl.

4. Виконайте команду ***ls***, не виходячи з редактора, переконайтеся, що файл testl створений.

5. Поставте курсор на початок 2-го рядка. Видаліть 2 символи.

6. Скопіюйте 3-й рядок і вставте його перед першим рядком.

7. Вставте вміст файлу testl на початок тексту.

8. Перейдіть на початок 2-го рядка.

9. Об'єднайте поточний і наступний рядки.

10. Завершіть роботу з редактором зі збереженням зроблених змін у файлі test2.

Варіант 5

1. Завантажтеся в систему під вашим користувацьким ім'ям.

2. Створіть новий текстовий файл text за допомогою редактора vi. Наберіть наведений нижче текст:

This is an example.

Test1, Test2, Test3

alpha / beta / gamma.

3. Збережіть набраний текст у файлі text.

4. Виконайте команду ***ls***, не виходячи з редактора, переконайтеся, що файл text створений.

5. Перейдіть на 1-й рядок. Об'єднайте поточний і наступний рядки.

6. Вставте вміст файлу text на кінець тексту.

7. Виріжте останній рядок і вставте його на початок тексту.

8. Поставте курсор перед словом beta в 2-му рядку та видаліть 3 символи.

9. Перейдіть на початок 1-го рядка.

10. Завершіть роботу з редактором без збереження зроблених змін.

Варіант 6

1. Завантажтеся в систему під вашим користувацьким ім'ям.

2. Створіть новий текстовий файл text за допомогою редактора vi. Наберіть два абзаци тексту. Текст повинен містити ваше прізвище (наприклад, у вигляді підпису). Запишіть файли під іменами text і text1, вийдіть із редактора.

3. Виконайте команду ***ls***, не виходячи з редактора, переконайтеся, що файл test1 створений.

4. Завантажте файл text у редактор, скопіюйте перші 2 рядки тексту в буфер і вставте їх на кінець тексту.

5. Запишіть файл під тим самим ім'ям.

6. Не виходячи з редактора, завантажте файл text1 і запишіть файл, не виходячи з редактора.

7. Користуючись пойменованими буферами, перенесіть 3 рядки тексту з першого файлу в другий. Збережіть зміни. Вийдіть з редактора.

8. Відкрийте у редакторі файл text і на кінець його додайте зміст файлу text1.

9. Запишіть отриманий файл як text2 і, не виходячи з редактора, видаліть файли text і text1.

10. Завершіть роботу з редактором без збереження зроблених змін.

Зміст звіту

1. Початкові дані та поставлення задачі.
2. Текст команд або поєднання клавіш для виконання кожного завдання.
3. Скріншоти, що підтверджують правильність виконання завдань.
4. Висновки.

Лабораторна робота 6

Робота з текстовими даними. Регулярні вирази в Linux

Мета – оволодіння практичними навичками перенаправлення стандартних потоків, роботи з фільтрами та організації конвеєрів.

Завдання для самостійної підготовки

1. Вивчити:
   - командні оболонки, їх запуск, конфігураційні файли;
   - стандартні потоки та їх перенаправлення;
   - організацію конвеєрів;
   - організацію фільтрів і команди, використовувані як фільтри.

2. Ознайомитися з такими командами UNIX:

tee, find, cut, date, grep, sort.

Звернути увагу на метасимволи *, ?, /, […], $ і на правила інтерпретації їх під час використання одинарних і подвійних лапок ‘…’ та "…". Розібратись із використанням у командах операторів перенаправлення потоків та організації конвеєрів ">", "<", "|" і використанням псевдопристрою */dev/null*.

3. Відповідно до завдання підготувати послідовність команд для його виконання.

Теоретичні відомості

У роботі буде розглянуто деякі можливості командних оболонок, які дозволяють користувачам гнучко формувати завдання для виконання операційною системою. Кожна командна оболонка в процесі свого запуску робить налаштовування системного оточення (виконує ініціалізацію системних змінних і змінних командної оболонки), для чого використовує визначені файли. Їх щонайменше два – глобальний і локальний (деякі оболонки можуть використовувати більшу кількість конфігураційних файлів). Глобальні конфігураційні файли для всіх оболонок розміщені в каталозі */etc*. Локальні конфігураційні файли для всіх оболонок

розміщені в домашньому каталозі користувача. Імена конфігураційних файлів зазвичай закінчуються на ‘rc’. Локальні файли роблять «прихованими», щоб вони не заважали користувачу в його повсякденній роботі (приховані файли мають ім’я, що починається з символу ‘.’, команда ls без параметрів їх не показує). Приклади конфігураційних файлів: для *sh – /etc/profile* і *~/.profile*, для *csh – /etc/cshrc* і *~/.cshrc*, для *tcsh – /etc/cshrc* і *~/.cshrc*, а також */etc/tcshrc* і *~/.tcshrc* (двох останніх може і не бути).

Часто як параметр деякої команди нам потрібно зазначити не один файл, а кілька файлів, назви яких мають певні спільні риси. В такому разі використовують так звані маски пошуку. Спеціальний символ ‘?’ в масці означає один будь-який символ, а спеціальний символ ‘*’ – будь-яку послідовність символів. Наприклад, команда *ls /bin/????* виведе на екран всі файли з каталогу */bin*, імена яких складаються з чотирьох символів, а команда *ls /etc/d** виведе на екран всі файли з каталогу */etc*, імена яких починаються з літери *d*. Також можна задати список символів, наприклад, маска *[abc]???* задає ім’я з чотирьох літер, перша з яких – *a, b* чи *c*.

Для пошуку файлів за певними ознаками можна використовувати команду *find*. Перший параметр цієї команди (обов’язковий) – це каталог, з якого починається пошук (наприклад, / – кореневий каталог), далі – параметр, що задає ознаку пошуку (наприклад, *-name* – пошук файлів, імена яких відповідають заданій масці, *–atime* – пошук файлів, дата модифікації яких відповідає заданій умові), далі іде власне маска пошуку, а далі – дія. Найтиповіша дія – *-print*, виведення результатів пошуку на екран. Якщо цей параметр не зазначити, пошук відбуватися буде, а от результатів його видно не буде.

Наприклад:

find / -name ".c" –print*

виведе на екран список усіх файлів, імена яких мають розширення .c, тобто вихідних кодів на мові програмування С5.

Фільтри

Фільтри – це команди, що мають вхідний і вихідний потоки.

Ця обставина дозволяє використовувати фільтри в конвейєрах. Можна будувати команди, що складаються з декількох фільтрів, зв'язаних конвейєром. Використання фільтрів полегшує написання програм.

Фільтри файлів

cat – вивести
more, *less* – вивести посторінково
head –n <*n*> – вивести перші n рядки
tail –n <*n* > – вивести останні n рядки
tee – роздвоєння виведення
wc – лічильник

Фільтри пошуку підрядка

Відмінність:
grep – пошук одного простого шаблону в групі файлів;
fgrep – декілька зразків одночасно в одному файлі *grep* використовує регулярний вираз , а *fgrep* – ні;
egrep – використовує в зразках розширений набір символів.

Найбільш поширений – *grep*
Синтаксис
grep <*зразок*> <*список файлів*>
<опції>:
-*i* – ігнорувати регістр;
-*c* – виводити лише кількість знайдених збігів;
-*l* – виводити лише імена файлів, що містять зразок;
-*n* – нумерує рядки;
-*v* – виводить рядки, що не містять зразок;
<*зразок*> : регулярний вираз із використанням символів:
* декілька;
. *?* один;
[ ] один із набору;

^ початок рядка;
$ кінець рядка.

Приклад

1. Створити файл students із списком групи. Підрахувати скільки в ньому рядків

cat students | wc –l.

2. Видати перші 10 прізвищ

head –n 10 students.

3. Видати останні 10 прізвищ

tail –n 10 students.

4. Знайти в ньому рядок, відповідний конкретному прізвищу

grep <fam> students.

5. Скільки прізвищ починається на d?

grep –c "^d" students

grep "^d" students | wc –l.

6. Знайти користувача user у файлі */etc/passwd*:

grep user /etc/passwd.

Редагувальні фільтри

sort – сортування;
diff – порівнює два файли та виводить результат порівняння;
uniq – з групи однакових вибирає один.

Опції *sort*

-o – записати у файл;
-c – перевіряє чи відсортований файл;
-u – виводить повторний рядок лише один раз;
-d – ігнорує символи, набір яких не є буквами, цифрами або пробілом;
-f – ігнорує регістр;
-r – зворотний порядок;
-b – ігнорує початкові пробіли;
+ <n> – пропускає n полів у рядку + 2 – почати з 3–го поля;
- <n> – закінчити на полі n;
- t – задає розділювач полів, за умовчанням пробіл.

Опції uniq

-c – перед кожним рядком ставить кількість збігів;

-d – виводить лише рядки, які повторюються;

-u – виводить лише ті рядки, які не повторюються;

-<n> – пропускає n полів у рядку;

+<n> – пропустити n полів + пробіли.

Для ефективного використання фільтра *uniq* файл заздалегідь потрібно сортувати.

Приклад

who | cut -f1 -d' ' | sort | uniq

Фільтри даних

cut – вирізати підрядок;

paste – вставити підрядок;

join – об'єднати рядки з різних файлів;

tr – заміна символів.

Опції cut

- f <n> номер поля, яке вирізається;

<n1>, <n2> група;

<n1> - <n2> – діапазон полів;

-c - <n1> <n2> – вирізається посимвольно;

-d <розділювач> – задає розділювач.

Приклад

Показати власників файлів у поточній директорії

ls –l | cut –f4 –d" "

Опції *tr <що замінити> <на що>*

[ ] – задає діапазон заміни;

-d – видаляє всі символи зі списку;

-c – замінює символ із першого списку символом із другого;

-s – замінює всі екземпляри набору символів, що провіряється, з першого списку одним екземпляром символів із другого.

Приклад
echo “aabcdefa” | tr – s "a" "z"

Ще одна можливість оболонок – перенаправлення потоків введення – виведення. Зазвичай більшість команд (утиліт) бере інформацію з клавіатури, або з файлу, якщо його зазначено як параметр, і виводить результати на екран. Проте фактично вони працюють із так званими стандартними потоками введення та виведення, які пов’язані з певними файлами. Файл у системі UNIX розглядається як потік байтів. Оскільки пристрої в UNIX розглядаються як файли, а операції введення та виведення – як читання та запис у відповідні файли, це дозволяє легко переводити вхідний і вихідний потоки з файлів на пристрої чи навпаки.

Стандартне введення за замовчанням зчитується з клавіатури. Якщо як параметр зазначено ім’я файлу, то замість стандартного введення відповідна утиліта буде організовувати вхідний потік із зазначеного файлу (але так діють не всі команди!). Вихідних потоків є два – стандартний потік виведення (за замовчуванням у сучасних системах – на екран монітору), і потік повідомлень про помилки (за замовчуванням – туди само).

Оболонка дає змогу перенаправити потоки в заданий файл. Символ ‘<‘ перенаправляє вхідний потік. Після цього символу очікується ім’я файлу або пристрою, з якого буде братися вхідний потік.

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

Команда *cat < my_file* на перший погляд буде робити те саме, але з точки зору операційної системи та самої утиліти *cat* все буде відбуватися по-іншому. В попередньому випадку командна оболонка запускала утиліту *cat* і передавала їй параметр командного рядка *my_file*, а сама утиліта *cat* вже

інтерпретувала цей параметр як ім'я вхідного файлу. В останньому випадку командна оболонка запускала утиліту *cat* без параметрів, але передавала їй вміст файлу *my_file* як вхідний потік.

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

Команда

cat < my_file1 > my_file2

перенаправляє як вхідний, так і вихідний потік. Якщо файл *my_file1* існує, то його вміст буде записано в файл *my_file2.* Якщо не існує, то повідомлення про помилку буде виведено на екран. Потік повідомлень про помилки в оболонці sh перенаправляється окремо, він позначається **2>**. Наприклад, команда

cat < my_file1 > my_file2 2> my_file3

у разі, якщо файл *my_file1* існує, запише його вміст у файл *my_file2*, а якщо не існує, то запише повідомлення про помилку в файл *my_file3*. Важливо наголосити, що оболонки *csh* і *tcsh* не мають засобів безпосередньо перенаправляти потік повідомлень про помилки. Проте засобами цих оболонок також можна організувати окреме перенаправлення потоків завдяки хитрим прийомам програмування. Інформацію про це можна знайти в літературних джерелах.

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

Перенаправлення введення – виведення широко використовується в двох випадках. Перший – це запуск утиліт у фоновому режимі. Щоб їхня робота не заважала роботі

користувача з терміналом, потрібно так перенаправити потоки введення – виведення, щоб вони працювали лише з файлами. Другий – це використання спеціальних команд-утиліт, які призначені саме для того, щоб прийняти певну інформацію з одного файлу, обробити її, а результат записати у другий файл. Такі утиліти називаються фільтрами. Утиліта *cat*, варіанти використання якої з перенаправленням потоків було розглянуто вище – це простіший фільтр. Він практично не обробляє інформацію, лише може зчіплювати кілька файлів в один. Інші корисні фільтри *cut, grep, sort.*

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

cut –d: –f1,5 < /etc/passwd.

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

Утиліта *grep* виводить лише ті рядки, в яких є заданий рядок пошуку. Утиліта *sort* виконує сортування вхідного потоку, наприклад, за абеткою. Докладніше про ці та інші фільтри можна дізнатися з довідкової системи *man*.

Існує можливість перенаправити вихідний потік однієї утиліти безпосередньо у вхідний потік іншої, без використання тимчасових файлів. Це так звані конвеєри (pipe). У системі UNIX всі утиліти, що поєднані в конвеєр, запускаються паралельно та обробляють інформацію за кількістю її надходження. Конвеєр утворюється за допомогою символу '|' так:

util1 | util2 | util3.

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

Приклад конвеєра:

ps –al | grep root | more.

Команда *ps* з ключами *–al* направить у вихідний потік список усіх процесів у системі, grep root вибере з них лише ті, які виконуються від імені root'а, more забезпечить виведення їх на екран посторінково. Інший приклад:

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

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

Якщо проміжні результати на якійсь із стадій конвеєра бажано зберегти, можна скористатися командою *tee my_file*. Ця команда візьме вхідний потік, передасть його без змін у вихідний потік і одночасно продублює у файл *my_file6*. Наприклад, так можна модифікувати один із розглянутих вище прикладів:

ps –ef | grep root | tee my_file | more.

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

Порядок виконання роботи
Завдання

1. Перейдіть у каталог */bin*. Перегляньте список усіх файлів, що починаються з символу (п.1), який визначено в таблиці індивідуальних завдань.

Таблиця 6.1 – Варіанти індивідуальних завдань

Варіант	**П. 1**	**П. 2**	**П. 3**	**П. 6, 7**
1	n	2	x, z, r, q	/usr/sbin
2	l	3	m, g, y	/sbin
3	k	5	a, d, k, l	/tmp (або /var/tmp)
4	h	4	x, y, z	Ваш домашній каталог
5	g	3	u, v, w	/
6	f	2	q, r, s, t	/var
7	d	5	m, n, o, p	/home
8	c	4	i, j, k, l	/usr/bin
9	b	3	e, f, g, h	/usr
10	a	2	a, b, c, d	/bin

2. Перегляньте список файлів, імена яких складаються з визначеної в таблиці індивідуальних завдань кількості символів (п. 2).

3. Перегляньте список файлів, імена яких починаються з символів, які визначено в таблиці індивідуальних завдань (п. 3). Зробіть це декількома способами.

4. Створіть у вашому домашньому каталозі підкаталог *lab_5* і перейдіть у нього.

5. За допомогою команди *cat* створіть файл *my_text* і запишіть у нього кілька рядків. Потім за допомогою команди *cat* допишіть у нього ще кілька рядків.

6. Підрахуйте кількість файлів у каталозі, визначеному з таблиці індивідуальних завдань, використовуючи і не використовуючи конвеєри. Порівняйте результат.

7. Підрахуйте кількість файлів у каталозі, визначеному з таблиці індивідуальних завдань, водночас зберігши список файлів у файлі *filelist*, використовуючи команду *tee*.

8. Виведіть на екран лише час, що повертається командою *date*.

9. Виведіть на екран список усіх користувачів системи, тобто перші поля кожного рядка файлу */etc/passwd* (роздільник полів – символ '**:**').

10. Виведіть на екран імена всіх файлів у каталозі */bin*, що містять слова «Software» чи «software». Потік помилок не повинний виводитися на екран.

Зміст звіту

1. Початкові дані та поставлення задачі.
2. Текст команд або поєднання клавіш для виконання кожного завдання.
3. Скріншоти, що підтверджують правильність виконання завдань.
4. Висновки.

Електронне навчальне видання

Методичні вказівки
до виконання лабораторних робіт
із дисципліни «**Мережеві операційні системи**»
для студентів спеціальності
172 *«Телекомунікації та радіотехніка»*
денної форми навчання

Частина 2

Відповідальний за випуск А. С. Опанасюк
Редактор Н. М. Мажуга
Комп'ютерне верстання О. В. Д'яченка

Формат 60x84/16. Ум. друк. арк. 2,79. Обл.-вид. арк. 2,44.

Видавець і виготовлювач
Сумський державний університет,
вул. Римського-Корсакова, 2, м. Суми, 40007
Свідоцтво суб'єкта видавничої справи ДК № 3062 від 17.12.2007.