**Робота №3. Командна оболонка shell, стандартні потоки введення/виведення, фільтри і конвеєри**

### Мета

**Оволодіння практичними навичками перенаправлення стандартних потоків, роботи з фільтрами і організації конвеєрів**

### Завдання для самостійної підготовки

1. Вивчити (довідковий матеріал і, наприклад, [2, пп. 1.4, 1.5]):

 командні оболонки, їх запуск, конфігураційні файли;

 стандартні потоки і їх перенаправлення;

 організацію конвеєрів;

 організацію фільтрів і команди, використовувані як фільтри.

1. Ознайомитись з такими командами UNIX: **tee, find, cut, date, grep, sort**

Звернути увагу на метасимволи **\*, ?, /, […], $** і на правила інтерпретації їх при використанні одинарних та подвійних лапок **'…'** та **"…".** Розібратись з використанням в командах операторів перенаправлення потоків і організації конвеєрів **">", "<", "|"** і використанням псевдопристрою **/dev/null**.

1. Відповідно до завдання підготувати послідовність команд для його виконання

## Довідковий матеріал

У цій роботі ми розглянемо деякі можливості командних оболонок, які дозволяють користувачам гнучко формувати завдання для виконання операційною системою.

Кожна командна оболонка в процесі свого запуску робить налаштування системного оточення (виконує ініціалізацію системних змінних та 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 знаходяться в каталозі /etc. Локальні конфігураційні файли для всіх оболонок знаходяться в домашньому каталозі користувача. Імена конфігураційних файлів, як правило, закінчуються на ‘rc’. Локальні файли роблять “прихованими”, щоби 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).

Конфігураційні файли є звичайними командними файлами (докладніше про командні файли див. Роботу №4).

Часто в якості параметру деякої команди нам треба вказати не один файл, а кілька файлів, назви яких мають певні спільні риси. В таких випадках 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, вивід результатів пошуку на екран. Якщо цей параметр не вказати, пошук відбуватись буде, а от результатів його видно не буде.

Наприклад:

### find / -name "\*.c" –print

виведе на екран список усіх файлів, імена яких мають розширення .с, тобто вихідних кодів на мові програмування С. [[1]](#footnote-1)

Ще одна можливість оболонок – перенаправлення потоків введення-виведення. Як правило, більшість команд (утиліт) приймає інформацію з клавіатури, або з файлу, якщо його вказано як параметр, і виводить результати на екран. Однак, фактично вони працюють із так званими стандартними потоками введення і виведення, які пов’язані з певними файлами. Файл в UNIX-подібних системах розглядається як потік байт. Оскільки пристрої в UNIX-подібних системах розглядаються як файли, а операції введення і виведення – як зчитування і записування у відповідні файли, це дозволяє легко переводити вхідний і вихідний потоки з файлів на пристрої чи навпаки.

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



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

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

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

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

### cat < my\_file1 > my\_file2

перенаправляє як вхідний, так і вихідний потік. Якщо файл my\_file1 існує, то його вміст буде записано у файл my\_file2. Якщо не існує, то повідомлення про помилку буде виведено на екран. Потік повідомлень про помилки в оболонці sh (і bash) перенаправляється окремо, він позначається **2>**. Наприклад, команда **cat < my\_file1 > my\_file2 2> my\_file3**

у разі, якщо файл my\_file1 існує, запише його вміст в файл my\_file2, а якщо не існує, то запише повідомлення про помилку в файл my\_file3. Важливо наголосити, що оболонки csh і tcsh (стандартні для систем BSD) не мають засобів безпосередньо перенаправляти потік повідомлень про помилки. Тим не менше, засобами цих оболонок також можна організувати окреме перенаправлення потоків завдяки хитрим прийомам програмування. Інформацію про це можна знайти в літературі.

Дуже часто потік помилок намагаються взагалі “загасити”. Для того, щоби знищити якийсь потік, існує спеціальний пристрій /dev/null. Все, що в нього направляється, зникає безслідно.

 Перенаправлення введення-виведення широко

використовується у двох випадках. Перший – це запуск утиліт у фоновому режимі (це буде розглянуто в Роботі №5). Щоби робота фонових утиліт не заважала роботі користувача з терміналом, слід так перенаправити потоки введеннявиведення, щоби вони працювали лише з файлами. Другий – це використання спеціальних команд-утиліт, які призначені саме для того, щоби 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 випадку ‘**:**’), ключ –f – список полів, що треба роздрукувати (у цьому випадку 1 і 5).

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

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

### util1 | util2 | util3

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

### ps –al | grep root | more

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

Інший приклад:

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

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

Якщо проміжні результати на деякій із стадій конвеєра бажано зберегти, можна скористатись командою tee my\_file. Ця команда візьме вхідний потік, передасть його без змін у вихідний потік і одночасно продублює у файл my\_file**[[2]](#footnote-2)**. Наприклад, так можна модифікувати один із розглянутих вище прикладів:

### ps –ef | grep root | tee my\_file | more

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

## Завдання до виконання

1. Перейдіть у каталог /bin. Перегляньте список усіх файлів, що починаються із символу, який визначено в таблиці індивідуальних завдань.
2. Перегляньте список файлів, імена яких складаються з визначеної у таблиці індивідуальних завдань кількості символів.
3. Перегляньте список файлів, імена яких починаються із символів, які визначено в таблиці індивідуальних завдань. Зробіть це декількома способами.
4. Створіть у вашому домашньому каталозі підкаталог lab\_4 і перейдіть в 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. **Таблиця індивідуальних завдань**

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

1. Підрахуйте кількість файлів у каталозі, визначеному з таблиці індивідуальних завдань, при цьому зберігши список файлів у файлі filelist, використовуючи команду tee.
2. Починаючи з вашого домашнього каталогу, виведіть на екран у повному форматі назви усіх файлів і каталогів, що починаються з ‘**m**’. При цьому перед виведенням кожної назви на екран повинен виводитися запит на його підтвердження.
3. Починаючи з кореневого каталогу, виведіть на екран імена всіх каталогів, що останній раз змінювалися більше 15 днів назад.
4. Виведіть на екран тільки час, що повертається командою date.
5. Виведіть на екран список усіх користувачів системи, тобто перші поля кожного рядка файлу /etc/passwd (роздільник полів — символ ‘**:**’).
6. Виведіть на екран імена усіх файлів у каталозі /bin, що містять слова Software чи software. Потік помилок при цьому не повинний виводитися на екран.

Увага!!! у цьому завданні мова йде про те, що слова

*Software* чи *software* містяться не у назві файлу (таких файлів там не повинно бути), а у самому файлі (а таких файлів має бути достатньо).

Відсортуйте конфігураційний файл вашої оболонки (.profile**,** .cshrc) відповідно до кодової таблиці ASCII так, щоб при цьому ігнорувалися пробіли на початку рядків. Робіть це з копією файлу, щоби не порушити нормальну працездатність вашої оболонки.

1. Якщо ваш Linux встановлено з вихідними кодами, то таких файлів буде безліч! [↑](#footnote-ref-1)
2. Походження назви цієї команди добре пояснює її дію. Взагалі “tee” – це назва літери “Т” в англійській абетці. Літера “Т” якраз і використовується, щоби проілюструвати відгалуження, розщеплення потоку на основний і бічний (коли натякають на розщеплення потоку на два еквівалентні, використовують літеру “Y”). [↑](#footnote-ref-2)