**Лекція № 6**

**2.2. Стандартні потоки даних у Linux**

Існує досить багато команд, які виводять на консоль певний результат. Типовим прикладом подібних команд є команда ***ls***. Виведення на консоль можна поділити на два типи: перший – результати роботи програми, другий – повідомлення про помилки, якщо команду виконати правильно не вдалось. Наприклад, подивившись на виведення команди ***ls***, можна побачити, що іноді вона може вивести і повідомлення про помилки.

Згідно з головною ідеєю операційної системи UNIX, яка проголошує, що все являє собою файл, команди насправді виводять свої результати в спеціальний файл, який називається **стандартним виведенням** (*standard output*, *stdout*), а повідомлення про помилки – в спеціальний файл **стандартне виведення помилок** (*standard error*, *stderr*). За замовчуванням обидва файли пов’язані з екраном і не зберігаються на диску. Крім того, існує багато команд, які потребують вхідних даних, які вони зазвичай оброблюють. Такі команди приймають введення зі спеціального файлу, який називається **стандартним введенням** (*standard input*, *stdin*), та за замовчуванням прив’язаний до клавіатури.

Зазвичай виведення здійснюється на екран, а введення – з клавіатури, але механізм перенаправлення введення/виведення операційної системи UNIX дозволяє змінити цей порядок.

**2.2.1. Перенаправлення стандартного виведення**

Механізм перенаправлення виведення дозволяє позначити, куди здійснюватиметься стандартне виведення. Для перенаправлення стандартного виведення не на екран а в інший файл використовується оператор ***>***, після чого зазначається його назва. Як приклад повідомимо командній оболонці, що вона має перенаправити результат команди ***ls*** у файл ***out.txt*** замість виведення його на екран. У прикладі використовується виведення вмісту каталогу ***/usr/bin***.



Давайте дослідимо перенаправлене виведення команди. Якщо вивести вміст файлу ***out.txt*** за допомогою команди ***less***, можна побачити, що у файлі дійсно збереглися результати роботи команди ***ls***:

***$ less out.txt***



Тепер повторимо експеримент з перенаправленням, але з невеликим ускладненням: вкажемо назву неіснуючого каталогу (навмисне допустимо помилку в шляху до каталогу – в каталозі ***/usr*** знаходиться каталог ***bin***, а не навпаки):



Оскільки команда з таким аргументом не може бути виконана нормальним чином, було отримано повідомлення про помилку. Як і багато типових команд UNIX, команда ***ls*** виводить повідомлення про помилки у стандартний потік виведення помилок. Оскільки ми вказали перенаправлення тільки стандартного потоку виведення, а не стандартного потоку виведення помилок, повідомлення про помилку було виведено на екран. Як перенаправити стандартний потік виведення помилок, буде показано трохи нижче, але перед цим подивимось, які зміни стались з файлом out.txt:



Як можна побачити, файл був повністю очищений. Причиною такого неочікуваного результату є те, що простий оператор перенаправлення без попередньої йому команди (у нашому випадку в стандартний потік виведення ніякої інформації не було виведено) створить новий порожній, якщо його до цього не існувало, або очистить існуючий. Для додавання виведення в кінець існуючого файлу без його перезапису використовується інший оператор перенаправлення – ***>>***:

**$ ls -l /usr/bin >> out.txt**

Оператор ***>>*** додає результати виконання команди в кінець файлу. Якщо файл не існує, він буде створений, як при використанні оператора ***>***.

**2.2.2. Перенаправлення стандартного виведення помилок**

Перенаправлення стандартного потоку виведення помилок виконується складніше, ніж стандартне виведення. Команда може виробляти виведення у будь-який з кількох нумерованих файлових потоків, однак перші з них представляють стандартні потоки введення, виведення і виведення помилок. Командна оболонка посилається на них у якості файлових дескрипторів **0**, **1** і **2** відповідно.

Таким чином можна перенаправляти будь-який потік у файл. Так перенаправити стандартний потік виведення у файл можна було б при використанні оператора ***1>***, однак самостійно оператор ***>*** (а разом з ним і ***>>***) за замовчуванням вказують про перенаправлення саме стандартного потоку виведення. Для перенаправлення стандартного потоку помилок в один файл потрібно використати оператор ***2>***. Наприклад:



Таким чином, був створений файл error.txt, у який була записана помилка виконання команди ls /bin/usr, яка полягає у тому, що каталогу, який вказаний у якості її аргументу, не існує. Переконаємось у цьому:

***$ less error.txt***



Дійсно, у файлі ***error.txt*** міститься повідомлення про помилку.

**2.2.3. Перенаправлення стандартного виведення і стандартного виведення помилок в один файл**

Іноді необхідно зберегти все виведення команди в одному файлі незалежно від того, чи вивела команда нормальний результат або завершилася помилково. Для досягнення цієї мети потрібно перенаправити одразу два потоки: стандартного виведення і стандартного виведення помилок. У якості прикладу виконаємо почергово вміст трьох каталогів: ***/sys***, ***/usr*** та ***/root***. Виведення вмісту перших двох каталогів буде виконано нормально, а третього завершиться помилкою, оскільки у звичайного користувача немає прав на читання змісту домашнього каталогу користувача **root**.



Таким чином, у представленому прикладі інформація була виведена як у стандартний потік виведення, так і у стандартний потік помилок. Існує два способи перенаправлення обох потоків в один файл. Перший з них є традиційним і використовує два перенаправлення: спочатку перенаправлення стандартного потоку виведення у файл звичайним способом, а потім перенаправлення стандартного потоку виведення помилок у стандартний потік виведення. Це виглядає наступним чином:

***$ ls /sys /usr /root > out.txt 2>&1***

Варто зауважити, що для перенаправлення одного стандартного потоку в інший використовується знак ***&***. Тепер переглянемо зміст отриманого файлу out.txt та переконаємося, що у ньому збереглася інформація з обох потоків:

***$ less out.txt***



Другий спосіб є більш сучасним та з’явився у більш нових версіях командного інтерпретатора **bash**. Замість двох перенаправлень у ньому використовується одне ***&>***, яке означає перенаправлення і стандартного потоку виведення і стандартного потоку виведення помилок в один і той самий файл:

***$ ls /sys /usr /root &> out.txt***

У Linux також існує можливість не виводити небажану інформацію, використовуючи спеціальний файл ***/dev/null***, куди можна перенаправити виведення. Файл представляє системний пристрій, що має назву **бітоприймач**, приймає будь-які вхідні дані та нічого з ними не робить. Наприклад для того, щоб приховати небажане виведення про помилки, достатньо здійснити наступне перенаправлення: ***$ ls -l /bin/usr 2> /dev/null***

**2.2.4. Команда cat**

Команда ***cat*** представляє собою типовий приклад команд, для роботи яких мають бути певні вхідні дані. За своїм визначенням ***cat*** читає вміст одного або декількох файлів, назви яких почергово передані їй у якості аргументів, та виводить їх вміст на екран, не виходячи при цьому з режиму командного рядка, на відміну від команди ***less***, що представляє програму посторінкового перегляду файлу, вихід з якої здійснюється натисканням клавіші ***q***. Іншими словами можна сказати, що команда ***cat*** копіює вміст файлу в сандартний потік виведення.

Як приклад, виведемо вміст файлу ***out.txt***, який уже був створений та розглядався у прикладах вище.



Оскільки команда ***cat*** не виходить з режиму командного рядка, то достатньо часто використовується для виведення вмісту коротких текстових файлів. Однак якщо разом з командою ***cat*** також використовуються оператори перенаправлення, то за допомогою цієї команди можна додатково зробити деякі корисні дії.

Як було зазначено вище, команді ***cat*** для роботи необіохідні вхідні дані. Зазвичай ми забезпечуємо її ними, вказуючи назви файлів, вміст яких потрібно вивести, у якості аргументів. Однак ця команда може бути викликана і без аргументів. У цьому випадку вона починає читати дані зі стандартного потоку введення, що буде означати очікування введення користувача певних даних з клавіатури.

Введення даних з клавіатури відбуватиметься допоки користувач не натисне комбінацію клавіш ***<CTRL>-<D>***, що означає досягнення кінця файлу на стандартному потоці введення.

У результаті команда ***cat*** продублює виведення рядків, які до того ввів користувач. Такий результат легко пояснити, оскільки команда спрацювала так, як було потрібно: вивела вміст файлу на стандартний потік виведення, тільки у якості файлу у цьому випадку виступав стандартний потік введення. Утім, такий спосіб виконання команди ***cat*** є досить безглуздим, оскільки ніякої практичної цінності її результат не представляє. Однак все змінюється, якщо використовувати її разом з операторами перенаправлення виведення.

Розглянемо наступний приклад. Виконаємо команду ***cat*** без аргументів, але з оператором перенаправлення стандартного потоку виведення у файл.

У цьому випадку команда не має аргументів, тому за вхідними даними знову звертається до стандартного потоку введення, тобто до клавіатури. Однак введені користувачем рядки не виводить, а записує їх у файл під назвою ***file1*** за допомогою оператора ***>***. Так само можна було б використати оператор перенаправлення ***>>***, якщо потрібно було б записати введені рядки в кінець файлу без втрати інформації, що можливо збереглася у відповідному файлі до цього. Щоб побачити результат, виконаємо команду ***cat*** звичайним способом, передавши їй у якості аргументу ***file1***:



Виконання команди ***cat*** у розглянутий вище спосіб представляє заповнення файлів даними «на льоту», не використовуючи жодного текстового редактора.

Розглянемо наступний приклад виконання команди ***cat***:

***cat file1 > file2***

У цьому випадку вхідні дані поступають до команди «нормальним» чином, оскільки у якості аргументу передається назва файлу. Якщо б не відбулося перенаправлення стандартного потоку виведення, вміст файлу був би виведений на екран, однак перенаправлення записує його вміст в інший файл, створюючи його копію. Таким чином, виконання команди cat описаним вище способом є абсолютною аналогією виконання команди: ***cp file1 file2***

У тому, що створена копія початкового файлу, можна переконатися, почергово розглянувши вміст першого та другого файлів:



Також команда ***cat*** може мати більше ніж один аргумент. У цьому випадку команда почергово виведе вміст кожного з файлів, які вказані у якості аргументів. Однак при використанні операторів перенаправлення вміст усіх файлів буде записаний в інший файл, який представлятиме результат їх об’єднання:

***cat file1 file2 > file3***

У наступному прикладі спочатку змінимо зміст ***file2***, щоб він не був повною копією ***file1***, а потім об’єднаємо вміст даних файлів у ***file3*** та переглянемо його:



Тепер, коли ми знаємо, що команда **cat** може приймати вхідні дані не тільки з файлів, назви яких зазначені у якості аргументів, але й зі стандартного потоку введення, виконаємо перенаправлення стандартного потоку введення шляхом використання оператора ***<***:



Використовуючи оператор перенаправлення ***<***, ми змінили джерело даних для стандартного потоку введення на файл ***file1***. Звернемо увагу, що був отриманий такий самий результат як і під час виклику команди ***cat*** з аргументом. Такий описаний спосіб немає переваг порівняно з передачею простого аргументу, але він демонструє, що файли можна використовувати у якості джерела даних для стандартного потоку введення. Існують інші команди, які у перенапраленні стандартного потоку введення знаходять краще застосування.

**2.2.5. Конвеєри та фільтри**

Можливості команди зчитувати дані зі стандартного потоку введення і виводити результати у стандартний потік виведення у UNIX також використовуються в одному з механізмів об’єднання команд, який називається **конвеєром**. Цей механізм можна впізнати за наявністю оператора **|** між двома командами:

**<команда1> | <команда2>**

Механізм роботи конвеєра полягає у перенаправлені результату виведення попередньої команди (***команда1***) у якості вхідних даних наступної (***команда2***). Таким способом можна сформувати цілий ланцюг команд, які передають результати своєї дії одна одній.

Зазвичай у конвеєрах використовують команди, які приймають вхідні дані, певним чином їх обробляють та виводять результат. Досить часто подібні команди називають **фільтрами**. Розглянемо деякі найбільш часто застосовувані фільтри. Однак детально ці команди будуть розглянуті у наступних розділах.

**Команда sort**

Як можна зрозуміти з назви, команда ***sort*** виконує сортування вхідних даних (за замовчуванням в алфавітному порядку). Подібно до команди ***cat***, у звичайному своєму застосуванні вона має аргумент або декілька, що представляють назви файлів, дані яких потрібно відсортувати, та виводить результат у стандартний потік виведення. Ті, хто тільки починає вивчення Linux зазначають, що при її використанні результат сортування зберігається, але це не так: команда ***sort*** виводить результат тільки у стандартний потік виведення, а початковий файл при цьому залишається таким, яким був. Власне ніщо не заважає перенаправити виведення команди ***sort*** у файл, однак і тут є нюанс: не можна перенаправляти вхідний файл у самого себе. ***sort file1 > file1* Неправильно!!!**

У цьому випадку вміст файлу буде очищений. Для коректного результату варто виконати перенаправлення результату в інший файл: ***sort file1 > file2* Правильно!**

Також це стосується і всіх інших команд-фільтрів.

Необов’язково вказувати команді вхідні дані у якості аргументів. За їх відсутності подібно команді ***cat***, команда ***sort*** почне приймати їх зі стандартного потоку виведення. Також цій команді можна передати вхідні дані конвеєром.

У якості прикладу виведемо назви всіх виконуваних файлів. Такі файли зберігаються у каталогах ***/bin*** та ***/usr/bin***.

***$ ls /bin /usr/bin | sort | less***



Припустимо, що ми хочемо вивести їх в алфавітному порядку, однак результат команди ***ls*** у цьому випадку складатиметься з двох відсортованих списків, по одному для кожного каталогу. Додавши в конвеєр команду ***sort*** можна отримати єдиний відсортований список. Наприкінці для подальшого перегляду в посторінковому режимі (список може виявитися доволі великим, щоб розміститися на екрані) виконаємо команду ***less***, передавши їй конвеєром результат команди ***sort***.

**Команда uniq**

Команда ***uniq*** призначена для пошуку та видалення повторюваних рядків. Зазвичай вона використовується разом з командою ***sort***, оскільки у якості вхідних даних приймає відсортований список даних з файлу, чи зі стандартного введення, або переданий конвеєром.

Команда ***uniq*** за замовчуванням виводить тільки унікальні рядки, видаляючи дублікати. Тому для гарантії відсутності повторень у списку виконуваних файлів з минулого прикладу, додамо ***uniq*** у конвеєр: ***$ ls /bin /usr/bin | sort | uniq | less***

Для того, щоб навпаки дізнатися, які назви виконуваних файлів є у каталогах /bin, та /usr/bin, виконаємо команду ***uniq*** з ключем ***-d***, який виводить список продубльованих рядків, але кожен рядок виводимо один раз:

***$ ls /bin/usr/bin | sort | uniq -d | less***



**Команда wc**

Команда **wc** (**Word Count** – лічильник слів) є спеціальним фільтром, який приймає вхідні дані та підраховує кількість рядків, слів і символів у файлах. Наприклад, виведемо відповідну статистичну інформацію щодо даних у файлі ***out.txt***:



У нашому прикладі було виведено три числа: кількість рядків, слів і символів (що також відповідають кількості байтів) у файлі ***out.txt***.

Відповідно до інших фільтрів, команда ***wc*** може викликатися без аргументів, приймаючи одночасно дані з клавіатури, або вхідні, які можуть бути передані їй конвеєром. Можна обмежити виведення команди за допомогою наступних ключів:

 -l – вивести тільки кількість рядків; -w – вивести тільки кількість слів; -с – вивести тільки кількість символів.

Наприклад, отримаємо кількість назв виконуваних файлів з відсортованого списку, що був отриманий у попередніх прикладах:

***$ ls /bin/usr /bin | sort | uniq | wc -l******2728***

**Команда grep**

Команда **grep** напевно є найбільш потужним фільтром, що використовується для пошуку рядків за шаблоном. Вона має наступний синтаксис:

***grep шаблон [файл ...]***

Обов’язковим аргументом для команди **grep** є тільки шаблон, усі інші аргументи представляють назви файлів, у вмісті яких потрібно шукати відповідні рядки і є необов’язковими (можна передати вхідні дані іншим чином). Коли команда **grep** знаходить у файлі збіг з шаблоном, вона виводить рядок зі знайденим збігом.

Для прикладу, серед назв виконуваних файлів знайдемо такі, які тим чи іншим чином пов’язані з архівацією файлів. Для простоти прикладу припустимо, що у назвах таких команд існує підрядок ***zip*** (насправді це не зовсім правильно). Зробити це можна наступним чином:



Команда **grep** має декілька достатньо зручних ключів:

***-i*** –ігнорувати регістр символів у процесі пошуку (за замовчуванням пошук виконується з урахуванням регістра символів);

***-v*** –навпаки виводить лише рядки, де збіг з шаблоном не знайдений.

**Команди head та tail**

Наприкінці розглянемо дві схожі команди, які виводять перші та останні рядки тобто – ***head*** та ***tail***. За замовчуванням обидві команди виводять до 10 рядків, однак їх кількість можна змінити, використавши разом з відповідними командами ключ ***-n***.

Для прикладу виведемо перші та останні 5 рядків файлу ***out.txt***:



Також команди ***head*** та ***tail*** можна використовувати в конвеєрах, що й демонструє наступний приклад:



У прикладі була виконана команда **last**, що відображає історію роботи у системі, однак за замовчуванням її результат представив би досить великий список, тому ми вивели тільки 5 найбільш свіжих записів, які у списку знаходяться на самому початку.