робота № 5

Потоки, програмні канали перенаправлення в ОС Linux

Мета: Дізнатися що таке потоки, як їх перенаправляти і розділяти, як перетворити потік в аргументи певної команди.

Завдання.

Створити директорію і файли для роботи з ними.
Перенаправити в файл стандартний вивід.
Перенаправити стандартний потік помилок.
Перенаправити два потоки в один файл.
Перенаправити вивід з використанням різних концепцій.
Розділити потоки.

Хід роботи

Для виконання прикладів в цій роботі ми будемо використовувати деякі файли. Створимо необхідну директорію і файли.

Для початку перейдіть в домашню директорію.

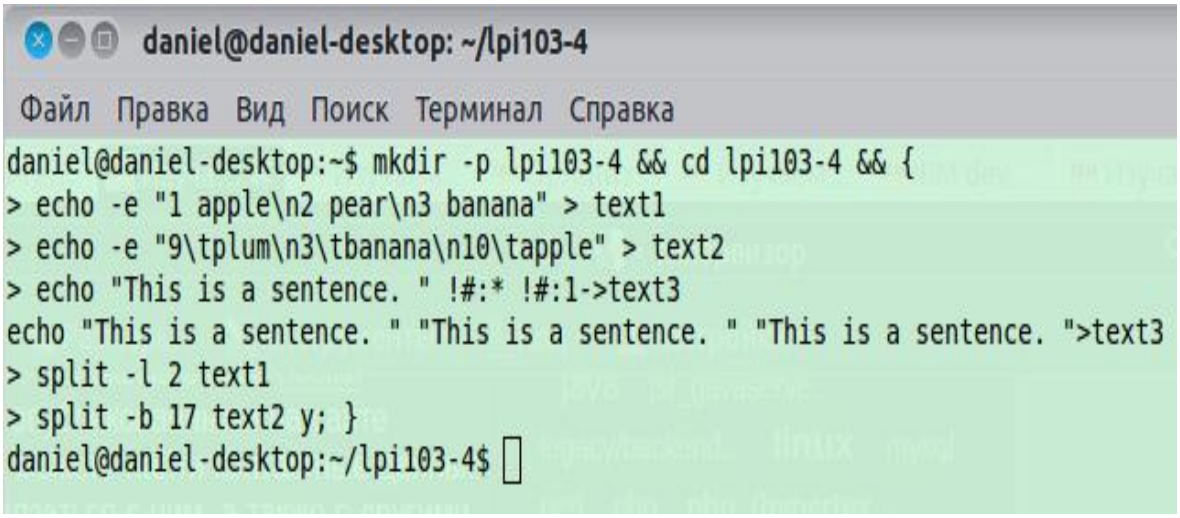
Відкрийте термінал і скопіюйте в нього вміст лістингу 1 (по рядкам, натискаючи *Ente*r після кожного рядка).

Лістинг 1. Створення файлів, необхідних для прикладів цієї роботи.

```
mkdir -p lpi103-4 && cd lpi103-4 && { echo
-e "1 apple\n2 pear\n3 banana" > text1
echo -e "9\tplum\n3\tbanana\n10\tapple" >
text2 echo "This is a sentence. " !#:* !#:1->text3
split -l 2 text1
split -b 17 text2 y; }
```

Ваше вікно має виглядати так, як показано в лістингу 2, а поточною директорією повинна стати щойно створена директорія *lpi103-4*.

Лістинг 2. Результати створення необхідних файлів.



Перенаправлення стандартного вводу/виводу.

Командний інтерпретатор Linux, такий як *bash,* отримує вхідні данні і направляє вихідні данні у вигляді послідовностей або потоків символів. Будь-який символ не залежить ні від попередніх, ні від наступних символів. Символи не впорядковані у вигляді структурованих записів або блоків з фіксованим розміром. Доступ до потоків здійснюється за допомогою механізмів вводу/виводу незалежно від того, звідки надходять і куди передаються потоки символів (файл, клавіатура, вікно, екран або інший пристрій вводу/виводу). Командні інтерпретатори Linux використовують три стандартні потоки вводу/виводу, кожному з яких назначений певний файловий дескриптор.

stdout – стандартний потік виводу, відображає вивід команд і має дескриптор;

stderr – стандартний потік помилок, відображає помилки команд і має дескриптор 2;

stdin – стандартний потік вводу, передає вхідні данні командам і має дескриптор 0.

Потоки вводу забезпечують передачу вхідних даних (зазвичай тих, що надходять з клавіатури) командам. Потоки виводу забезпечують друк текстових символів, як правило, на термінал. Спершу термінал являв собою ASCII друкуючий пристрій або дисплейний термінал, але зараз, як правило, це просто вікно робочого столу комп'ютера.

Перенаправлення виводу.

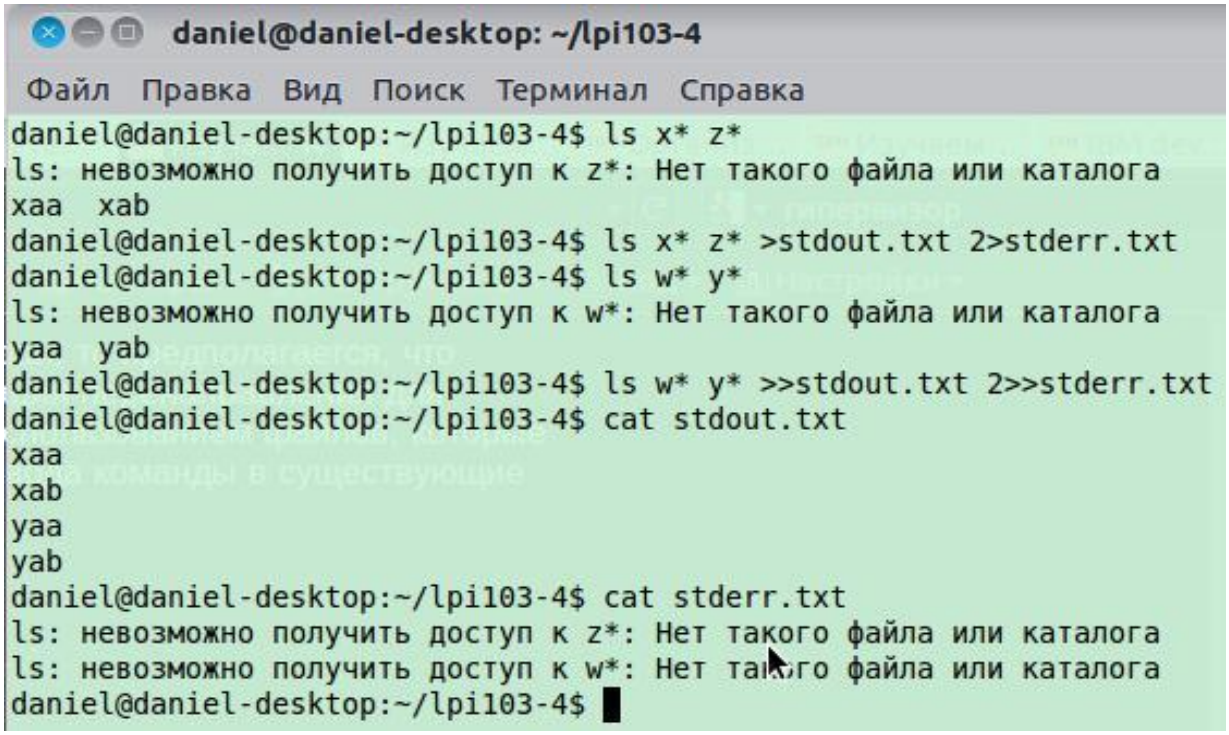
Існує два способи перенаправлення виводів в файл:

n> – перенаправляє вивід із файлового дескриптора n в файл. Ви повинні мати права на запис в файл. Якщо файл не існує, то він буде створений. Якщо файл існує, то весь його вміст, як правило, знищується без яких-небудь попереджень.

n>> – також перенаправляє вивід із файлового дескриптора *n* в файл. Ви також повинні мати права на запис в файл. Якщо файлу не існує, то він буде створений. Якщо файл існує, то вивід додається до його вмісту.

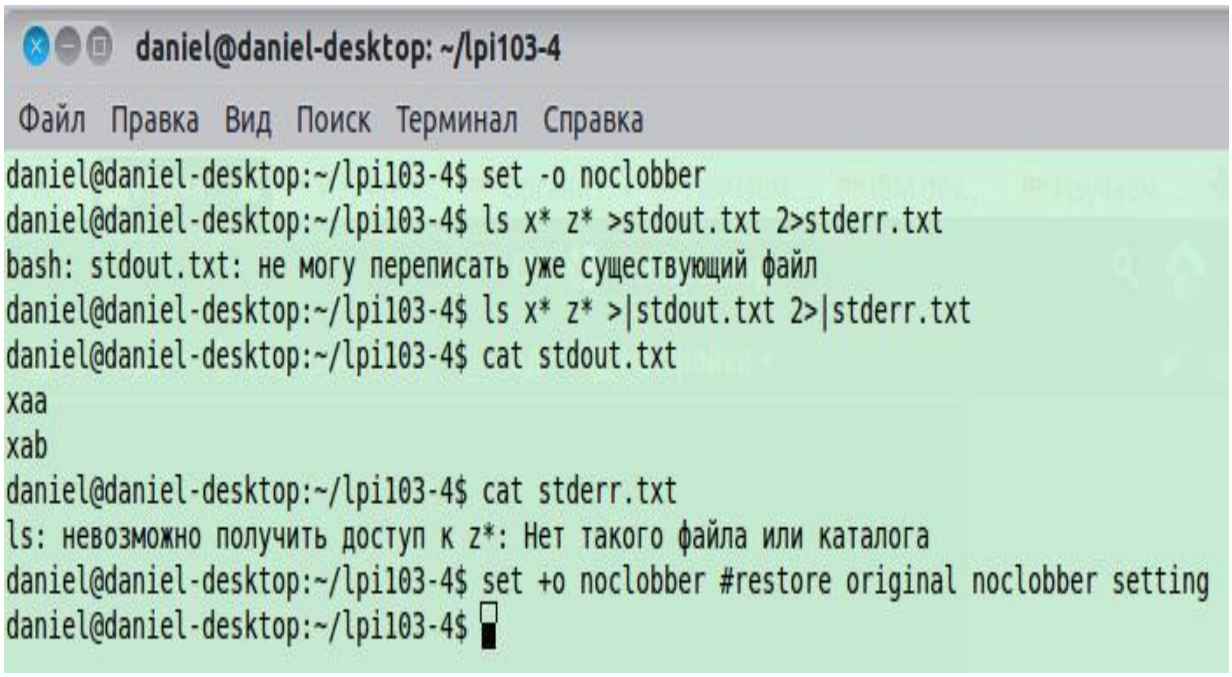
Символ *n* в операторах *n* > або *n* >> є файловим дескриптором. Якщо він не вказаний, то припускається, що використовується стандартний пристрій виводу. В лістингу 3 продемонстрована операція перенаправлення для розділення стандартного потоку виводу і стандартного потоку помилок команди *ls* з використанням файлів, які були створені раніше в директорії *lpi103-4*. Також продемонстровано додавання виводу команди в існуючі файли.

Лістинг 3. Перенаправлення виводу.



Перенаправлення виводу за допомогою оператора *n*> зазвичай призводить до перезапису існуючих файлів. Ви можете керувати цією властивістю за допомогою опції *noclobber* вбудованої команди *set*. Якщо ця опція визначена, то ви можете перевизначити її за допомогою оператора n>|, як показано в лістингу 4.

Лістинг 4. Перенаправлення виводів за допомогою опції noclobber.

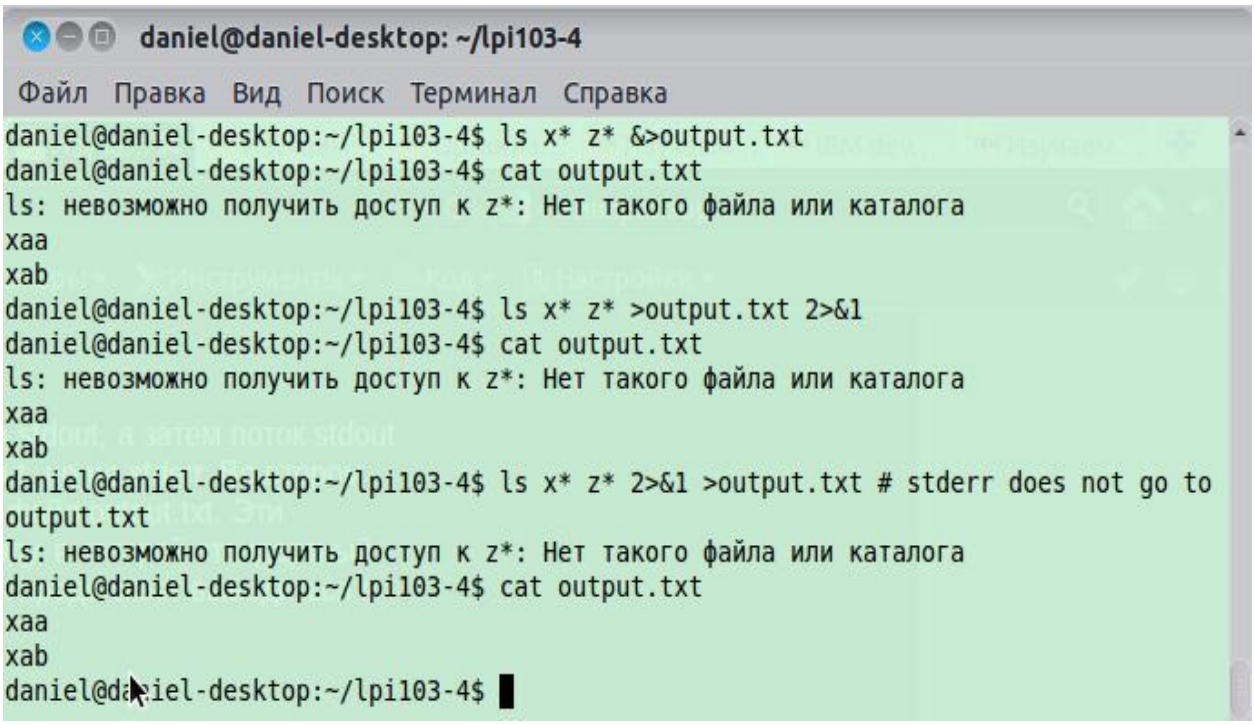


Іноді може знадобитися перенаправити в файл як стандартний вивід, так і стандартний потік помилок. Часто це використовується в автоматизованих процесах або фонових завданнях для того, щоб потім можна було подивитися результати роботи. Щоб перенаправити стандартний вивід і стандартний потік помилок в одне і те ж місце, використовуйте оператор *&>* або *&>>*. Альтернативний варіант – перенаправити файловий дескриптор *n* і потім файловий дескриптор *m* в одне і те ж місце за допомогою конструкції *m>&n* або *m>>&n*. В цьому випадку важливий порядок перенаправлення потоків. Наприклад, команда *command 2>&1 >output.txt* це не те ж саме, що команда *command >output.txt 2>&1*.

першому випадку потік помилок *stderr* перенаправляється в поточне місце розташування потоку *stdout*, а потім потік *stdout* перенаправляється в файл output.txt; однак друге перенаправлення зачіпає лише *stdout*, але не *stderr*.

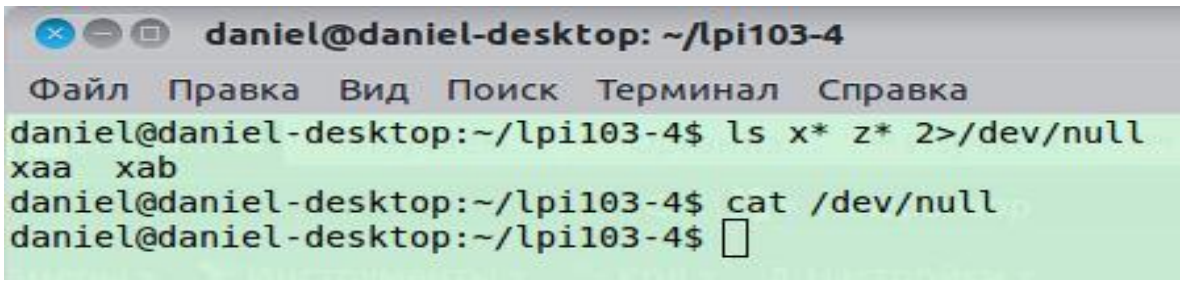
другому випадку потік *stderr* перенаправляється в поточне місце розташування потоку *stdout*, тобто, в файл output.txt. Ці перенаправлення проілюстровані в лістингу 5. Зверніть увагу на останню команду, в якій стандартний вивід був перенаправлений після стандартного потоку помилок, і, як наслідок, потік помилок продовжує виводитися у вікно терміналу.

Лістинг 5. Перенаправлення двох потоків в один файл.



інших випадках вам може знадобитися повністю проігнорувати стандартний вивід або стандартний потік помилок. Для цього слід перенаправити відповідний потік в пустий файл */dev/null*. В лістингу 6 показано, як проігнорувати потік помилок команди ls і як за допомогою команди cat переконатися в тому, що файл */dev/null* насправді пустий.

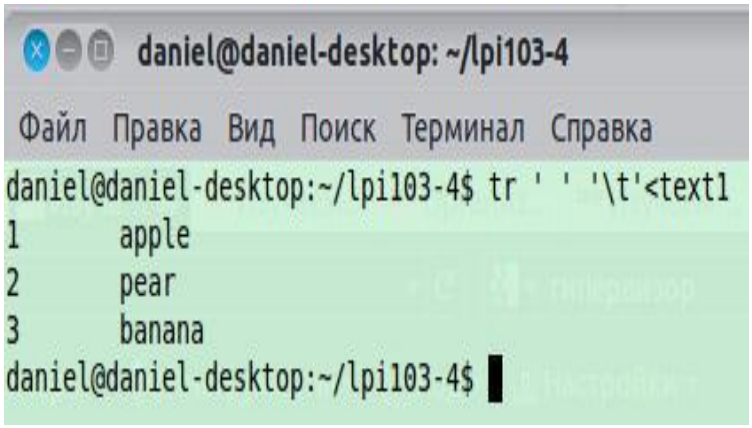
Лістинг 6. Ігнорування стандартного потоку помилок шляхом використання */dev/null.*



Перенаправлення вводу.

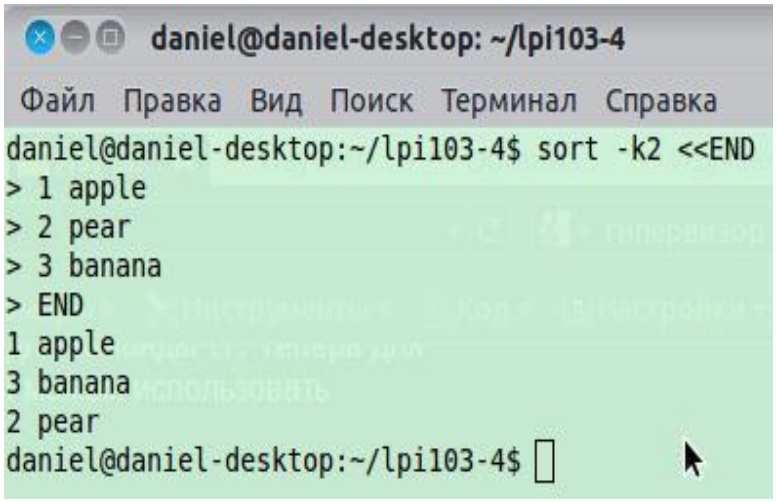
Таким же чином, як ми можемо перенаправляти потоки *stdout* і *stderr*, ми можемо перенаправити потік *stdin* із файлу за допомогою оператора <. Команда *tr* використовується для заміни пробілів в файлі *text1* на символи табуляції. Вивід команди *cat* можна використати для того, щоб створити стандартний потік вводу для команди *tr*. Але для перетворення пробілів в символи табуляції замість даремного виклику команди *cat* ми можемо використати перенаправлення вводу, як показано в лістингу 7.

Лістинг 7. Перенаправлення вводу.



В командних інтерпретаторах, в тому ж числі і в *bash*, реалізована концепція *here-document*, яка є одним із способів перенаправлення вводу. В ній використовується конструкція << і будь-яке слово, наприклад *END*, яке є маркером, або сигнальною міткою, що означає кінець вводу. Ця концепція продемонстрована в лістингу 8.

Лістинг 8. Перенаправлення вводу з використанням концепції here-document.



Але чому просто не можна набрати команду *sort -k2*, ввести данні і натиснути комбінацію *Ctrl-d*, яка позначає кінець вводу? Зрозуміло, що ви могли б виконати цю команду, але тоді ви не дізналися б про концепцію *here-document*, яка дуже часто використовується сценаріях командною оболонкою (в яких не існує іншої можливості вказати, які саме рядки повинні сприйматися в якості вводу). Оскільки для вирівнювання тексту і забезпечення зручності читання в сценаріях використовуються символи табуляції, то існує інший прийом використання концепції *here-document*. При використанні оператора << – замість оператора << початкові символи табуляції видаляються.

лістингу 9 ми використовували підстановку команд для створення символу табуляції, а потім створили невеликий сценарій командної оболонки, що містить дві команди *cat*, кожна з яких зчитує данні з блоку *here-document*. Зауважте, що ми використовували слово *END* в якості сигнальної мітки блоку *here-document*, який ми зчитуємо з терміналу. Якби ми використовували це ж саме слово в нашому сценарії, то наш ввід закінчився би передчасно. Тому замість слова *END* ми використовуємо в сценарії слово EOF. Після того, як наш

сценарій створений, ми використовуємо команду **.** (точка) щоб запустити його в контексті поточного командного інтерпретатора.

Лістинг 9. Перенаправлення вводу з використанням концепції here-document.

Розділення виводу.

Іноді може виникнути необхідність продивлятися вивід на екрані і одночасно зберігати його в файл. Для цього ви могли б перенаправити вивід команди в файл в одному вікні, а потім за допомогою *tail -fn1* відслідковувати виводи в іншому вікні, однак простіше за все використовувати команду *tee*.

Команда *tee* використовується в конвеєрі, а її аргументом є ім'я файлу (або імена декількох файлів), в який буде передаватися стандартний вивід. Опція *-a* дозволяє не заміщати старий вміст файлу новим вмістом, а додавати данні в кінець файлу. Якщо ви хочете зберегти як стандартний вивід, так і потік помилок, то необхідно перенаправити потік *stderr* в потік *stdout* перш, ніж передавати данні на вхід команді *tee*. В лістингу 10 наведений приклад використання команди *tee* для збереження виводу в два файли, *f1 і f2*.

Лістинг 10. Розділення потоку *stdout* за допомогою команди *tee*.

```
daniel@daniel-desktop:~/lpi103-4$ ls text[1-3]|tee f1 f2
text1
text2
text3
daniel@daniel-desktop:~/lpi103-4$ cat f1
text1
text2
text3
daniel@daniel-desktop:~/lpi103-4$ cat f2
text1
text2
text3
daniel@daniel-desktop:~/lpi103-4$
```

Контрольні запитання

Дайте визначення потокам. Що таке багатопотоковість?

Які основні операції з потоками?

Назвіть і опишіть способи реалізації потоків?

Які елементи сумісно використовуються потоками?

Які елементи окремо використовуються потоками?