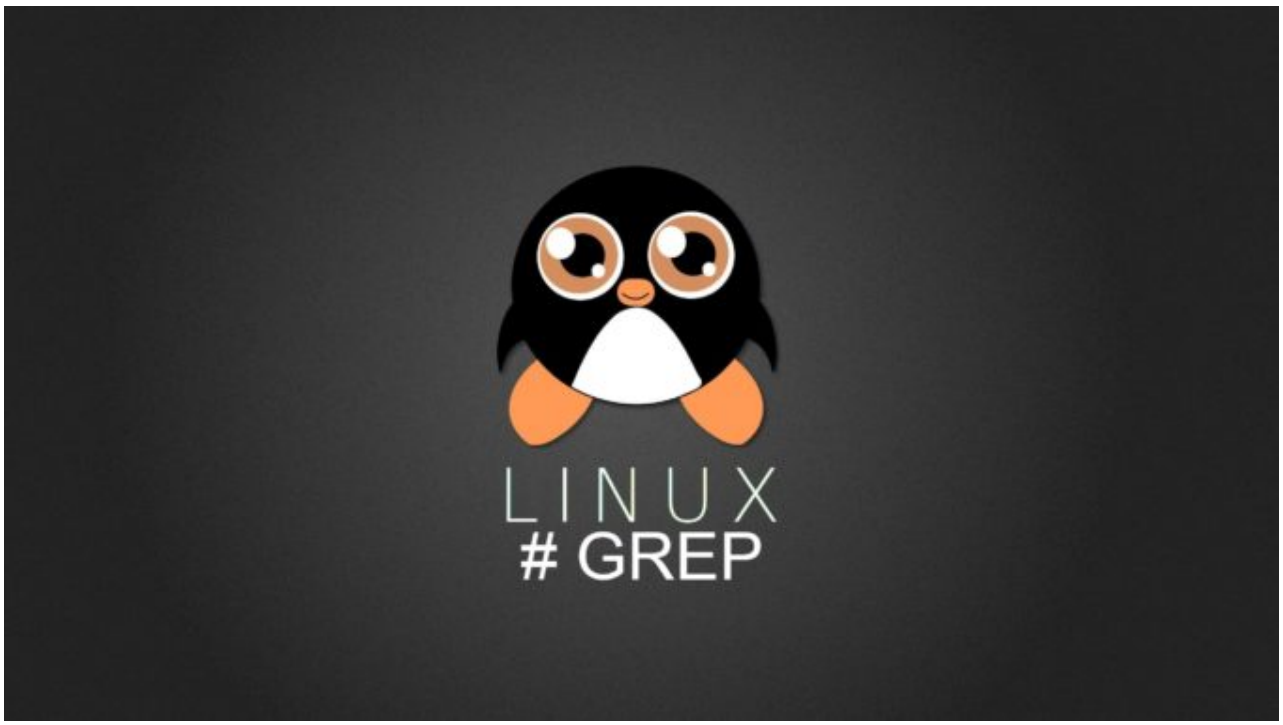
12 практичних прикладів команди grep в Linux

cinex.ho.ua/12-praktychnyh-prykladiv-komandy-grep-v-linux

Cinex 16 квітня 2019 р.



Ви коли-небудь стикалися з завданням пошуку певного рядка або шаблону у файлі, але й гадки не мали, з чого почати цей пошук? Ну тоді команда **grep** прийде до вас на допомогу!

Grep - потужний файлова пошукова система, яка поставляється в кожному дистрибутиві **Linux**. Якщо з якої-небудь причини він не встановлений у вашій системі, ви можете легко встановити його через менеджер пакетів (apt-get на Debian / Ubuntu і yum на RHEL / CentOS / Fedora).

Shell

```
1  sudo apt-get install grep <strong>#Debian/Ubuntu</strong>
2  sudo yum install grep <strong>#RHEL/CentOS/Fedora</strong>
```

Ми виявили, що найпростіший спосіб зрозуміти **grep** - це просто зануритися в нього з головою і спробувати реалізувати деякі приклади.

1. Пошук файлів

Припустимо, що ви тільки що встановили нову версію **Ubuntu** на ваш комп'ютер і що ви збираєтеся додати до **Python** деякі скрипти. Ви проводите пошук в Інтернеті, і бачите, що використовуються дві різні версії **Python**, і ви не знаєте,

який з них був встановлений у вашій системі монтажником **Ubuntu**, і встановлював він які-небудь додаткові модулі. Щоб з цим розібратися, просто запустіть цю команду:

```
1  dpkg -l | grep -i python
```

приклад виведення:

```
1  ii python2.7 2.7.3-0ubuntu3.4 Interactive high-level object-oriented language
2  (version 2.7)
3  ii python2.7-minimal 2.7.3-0ubuntu3.4 Minimal subset of the Python language
4  (version 2.7)
   ii python-openssl 0.12-1ubuntu2.1 Python wrapper around the OpenSSL library
   ii python-pam 0.4.2-12.2ubuntu4 A Python interface to the PAM library
```

Спочатку ми запустили **dpkg -l**, щоб вивести встановлені ***.deb** пакети у вашій системі. Далі ми передали цей висновок в **grep -i python**. Опція **-i** використовується, щоб ігнорувати **-case**, оскільки **grep** чутливий до регістру. Використання опції **-i** - хороша звичка, якщо ви звичайно, не намагаєтеся виконати більш конкретний пошук.

2. Пошук і фільтрація файлів

Grep також може використовуватися для пошуку і фільтрації усередині одного або декількох файлів. Давайте розглянемо цю функцію:

У вас виникли проблеми з вашим веб-сервером **Apache**, і ви звернулися до одного з багатьох форумів в мережі з проханням про допомогу. Добра душа, яка вам відповіла, попросила вас опублікувати вміст вашого файлу/etc/apache2/sites-available/default-ssl. Хіба вам не було б не легше, якби ви могли просто видалити всі прокоментовані рядки? Але ж ви це можете! Просто виконайте цю команду:

```
1  grep –v “#”  /etc/apache2/sites-available/default-ssl
```

Опція **-v** вказує **grep** інвертувати свій висновок, що означає, що замість друку співпадаючих рядків **grep** зробить протилежне і виведуться рядки, які не відповідають висловом, в цьому випадку - прокоментовані рядки.

3. Знайти всі файли .mp3

Grep може бути дуже корисний для фільтрації з **stdout**. Наприклад, припустимо, що у вас є ціла папка, повна музичних файлів різних форматів. Ви хочете знайти всі файли ***.mp3** у виконавця **JayZ**, але ви не хочете ніяких ремікс-треків. Використання команди **find** з декількома опціями **grep** запросто виконає цей трюк:

```
1  find . –name “*.mp3” | grep –i JayZ | grep –vi “remix”
```

У цьому прикладі ми використовуємо **find** для виведення всіх файлів з розширенням ***.mp3**, пов'язуючи пошук з **grep -i**, щоб відфільтрувати і вивести всі файли з ім'ям «**JayZ**», а потім ще одна опція для **grep -vi**, яка фільтрує і не виводить всі імена файлів з рядком «**remix**».

4. Відображення кількості рядків до або після рядка пошуку

Ще пара опцій - це прапорці **-A** і **-B**, які відображають узгоджені рядки і кількість рядків, які присутні до або після рядка пошуку. У той час як довідкова сторінка дає більш докладне пояснення, нам легше запам'ятати параметри як **-A** = **after** і **-B** = **before**:

```
1  ifconfig | grep –A 4 eth0
2  ifconfig | grep -B 2 UP
```

5. Вивід кількості рядків збігу

Параметр **-C** для **grep** замість того, щоб перевіряти рядки, які з'являються до або після шуканого рядка, він перевіряє рядки в будь-якому напрямку:

```
1  ifconfig | grep –C 2 lo
```

6. Кількість збігів

Grep так само може запросто підрахувати кількість збігів:

```
1  ifconfig | grep –c inet6
```

7. Пошук файлів за заданими номерами рядків

Параметр **-n** для **grep** дуже корисний при налагодженні файлів під час компіляції. Він відображає номер рядка у файлі пошуку:

```
1  grep –n “main” setup..py
```

8. Пошук рекурсивної рядку у всіх каталогах

Якщо ви хочете знайти рядок у поточному каталозі разом з усіма підкаталогами, ви можете вказати параметр **-r** для пошуку рекурсивно:

```
1  grep –r “function” *
```

9. Пошук по всьому шаблоном

Опції **-w** для **grep** шукає весь шаблон, який знаходиться в рядку. Наприклад, використовуючи:

```
1  ifconfig | grep –w “RUNNING”
```

Виведе рядок, що містить шаблон в лапках. З іншого боку, якщо ви спробуєте:

```
1  ifconfig | grep –w “RUN”
```

Нічого не буде повернуто, оскільки ми шукаємо НЕ шаблон, а ціле слово.

10. Пошук рядка в архівах файлах Gzip

Gzip (скорочення від GNU Zip) - утиліта стиснення і відновлення (декомпресії) файлів, що використовує алгоритм Deflate. Для пошуку в таких файлах необхідно використовувати команду **zgrep**, яка приймає ті ж параметри, що і **grep**, і використовується таким же чином:

```
1  zgrep –i error /var/log/syslog.2.gz
```

11. Пошук відповідності регулярному виразу в файлах.

Egrep - це ще одна похідна, яка означає «Розширений глобальний регулярний вираз». Вона розпізнає додаткові метасимволи-вирази, такі як **+? |** а також **()**.

Egrep дуже корисний для пошуку вихідних файлів і іншої частин коду, якщо виникне така необхідність. Вона може бути викликана з регулярного **grep**, вказавши параметр **-E**.

```
1  grep -E
```

12. Пошук рядків по фіксованому шаблону

Fgrep шукає файл або список файлів для фіксованого рядку шаблону. Це те ж саме, що і **grep -F**. Звичайний спосіб використовувати **fgrep** - передати йому файл шаблонів:

```
1  fgrep –f file_full_of_patterns.txt file_to_search.txt
```

***Додатково*:**

Використання `grep` в чистому виді

```
1  grep '12:00' /home/david/backup/log.txt
```

Ця команда показує як можна використовувати **grep** для того щоб отримати рядки з файлу, які містять підрядок вказаний в командному рядку. Файл не обов'язково повинен закінчуватися на .txt. Показання вище команда здійснює

пошук підрядка 12.00 в файлі /home/david/backup/log.txt і відображає всі рядки де ця подстрока зустрічається.

Ця комбінація може бути використана наприклад для пошуку бекапів які відбувалися о 12:00.

```
1  grep -v '12.00' /home/david/backup/log.txt
```

А ось ця команда (з використанням ключа **-v**) навпаки покаже тільки ті рядки де підрядок '12:00 'не зустрічається.

```
1  grep -l 'delay' /code/*.c
```

Ця команда буде шукати всі файли, що закінчуються на **.c** і текст в знайдених файлах відповідний підрядку '**delay**' і в кінцевому підсумку виведе тільки імена файлів де ця подстрока зустрічається.

```
1  grep -w '\<bay' * $ grep -w 'watch\>' *
```

Ця команда вже більш складна і складається з комбінації двох команд **grep**. Перша шукає рядки які починаються зі слова '**bay**' а друга рядки які закінчуються на слово '**watch**'.

Використання `grep` разом з потоками

```
1  ls -l | grep rwxrwxrwx
```

Ви напевно вже знаєте що команда **ls -l** відображає докладний список файлів в директорії. Частина **grep rwxrwxrwx** фільтрує результат отриманий від **ls -l** і виводить тільки ті директорії у яких встановлені відповідні права доступу. В даному випадку це відкритий доступ на читання, запис і пошук для всіх користувачів і груп. Так що замість того, щоб побачити повний список файлів ви побачите тільки ті файли у яких встановлені потрібні вам права доступу.

Висновок від команди **grep** може також бути спрямований потоком в іншу команду, наприклад як в наступному прикладі:

```
1  grep '^#' /home/david/script1 | more
```

Ця команда відобразить всі рядки в файлі /home/david/script1 які починаються з символу '#'. Визначення тип '^ #' означає що символ '#' повинен бути першим символом в строці.

```
1  grep -v '^[0-9]' /home/david/backup/log.txt | more
```

Ця команда шукає рядки містять в першому символі цифри від 0 до 9, а потім виводить тільки ті рядки які не були в результати пошуку. Як ви бачите - був використаний ключ '-v' означає реверсивний пошук.

Важливо: Необхідно укладати шукані рядки в одинарні лапки, як зазначено в двох попередніх прикладах для того, щоб інтерпретатор командного рядка міг сприймати їх коректно. Інакше інтерпретатор може зрозуміти це як іншу команду і результат виконання буде непередбачуваний.

Деякі додаткові ключі команди `grep` :

- `-v` : Виводь реверсивні результати. Замість того щоб вивести рядки де шукане було знайдено - виводь ті рядки де шуканої підстроки немає.
- `-c` : Відключає стандартний спосіб виведення результату і замість цього відображає тільки число позначає кількість знайдених рядків.
- `-i` : Робить пошук регістронезалежним
- `-w` : Веде пошук по цільним словами. Наприклад при звичайному пошуку рядка 'wood' grep може знайти слово 'hollywood'. А якщо використовується даний ключ то будуть знайдені тільки рядки де є слово 'wood'.
- `-l` : Виводить тільки імена файлів де був знайдений рядок.
- `-r` : Виробляє пошук рекурсивно по всіх піддиректоріях.

Це всього лише відправна точка для команди **grep**, як ви бачите, вона може впоратися з самими різноманітними завданнями. Крім простих команд реалізованих в один рядок, **grep** можна використовувати для написання потужних завдань **cron** і надійних сценаріїв оболонки.

Будьте винахідливі, експериментуйте з варіантами зі сторінок керівництва і придумуйте вираження **grep**, які послужать вашим власним цілям!