

Linux command line

1 cut

```
$cut -f2 animals.txt    # - вырезать поле в каждой строке
$cut -f1,3 animals.txt  # - вырезать несколько полей
$cut -f2-4 animals.txt  # - указав диапазон
```

Вывести все имена пользователей и отсортировать их:

```
$cut -d: -f1 /etc/passwd | sort
```

```
$cut -c1-3 animals.txt  # - по положению символа в строке
```

2 date

```
$date +%Y-%m-%d    # - Формат год-месяц-день: 2021-06-28
$date +%H:%M:%S    # - Формат часы:минуты:секунды 16:57:33
$date +"it's already %A!"    # - it's already Tuesday!
```

3 seq

```
$seq 1 5    # - Выводит все целые числа от 1 до 5 включительно
$seq 1 2 10  # - Увеличение на 2 вместо 1
$seq 3 -1 0   # - отрицательный шаг
$seq 1.1 0.1 2    # - Увеличение на 0,1
$seq -s/ 1 5    # - Разделение значений с помощью косой черты
$seq -w 8 10    # - приводит все значения к одинаковой ширине
```

4 Расширение команд с помощью фигурных скобок

```
$echo {1..10}    # - Вперед, начиная с 1: 1 2 3 4 5 6 7 8 9 10
$echo {10..1}    # - Назад, начиная с 10
$echo {01..10}   # - С ведущими нулями (для равной ширины)
```

```
$echo {1..1000..100}    # - Приращение сотнями, начиная с 1
$echo {1000..1..100}    # - Уменьшение сотнями, начиная с 1000
$echo {01..1000..100}    # - С ведущими нулями
```

Фигурные скобки vs квадратные:

```
$ls file[2-4]    # - Соответствует существующим именам файлов
$ls file{2..4}    # - Вычисляется в: file2 file3 file4
```

```
$echo {A..Z}    # - A B C ... X Y Z
$echo {A..Z} | tr -d ' '    # - Удалить пробелы: ABC...XYZ
```

5 find

```
$find /etc -print    # - Список всех каталогов в /etc рекурсивно
$find . -type f -print    # - Только файлы
$find . -type d -print    # - Только каталоги
```

Файлы, заканчивающиеся на .conf:

```
$find /etc -type f -name "*.conf" -print
```

Шаблон нечувствительный к регистру:

```
$find . -iname "*.txt" -print
```

Выполнить команду для каждого найденного файла, в конце обязательно ";" или экранированную \;

```
$find /etc -exec echo @ {} @ ";"
$find /etc -type f -name "*.conf" -exec ls -l {} " "
```

6 yes

`$yes` # - Выводит «у» по умолчанию

`$yes woof!` # - Повторять любую другую строку

7 grep

`$grep his frost` # - Вывести строки, содержащие «his»

`$grep -w his frost` # - Искать точное соответствие «his»

`$grep -i his frost` # - игнорировать регистр букв

`$grep -l his *` # - В каком файле содержится «his»?

Соответствие	Используемые выражения	Пример
Начало строки	<code>^</code>	<code>^a</code> = строка, начинающаяся с a
Конец строки	<code>\$</code>	<code>!\$</code> = строка, заканчивающаяся восклицательным знаком
Любой одиночный символ (кроме новой строки)	<code>.</code>	<code>...</code> = любые три последовательных символа
Знаки вставки, доллара или любой другой специальный символ c	<code>\c</code>	<code>\$</code> = знак доллара
Ноль или более вхождений выражения E	<code>E*</code>	<code>_*</code> = ноль или более знаков подчеркивания
Любой одиночный символ в наборе	<code>[characters]</code>	<code>[aeiouAEIOU]</code> = любая гласная
Любой одиночный символ, не входящий в набор	<code>[^characters]</code>	<code>[^aeiouAEIOU]</code> = любая негласная
Любой символ в диапазоне между c1 и c2	<code>[c1-c2]</code>	<code>[0-9]</code> = любая цифра
Любой символ вне диапазона между c1 и c2	<code>[^c1-c2]</code>	<code>[^0-9]</code> = любой нецифровой символ
Любое из двух выражений E1 или E2	<code>E1\ E2</code> для grep и sed, E1/E2 для awk	<code>one two</code> = или one, или two
Группировка выражения E с учетом приоритета	<code>\(E\)</code> для grep и sed, (E) для awk	<code>\(one\ two\)*</code> <code>(one two)*</code> = ноль или более вхождений one или two

`$grep -v '^$' myfile` # - все непустые строки, -v - исключает

`$grep 'cookie\|cake' myfile` # - содержащие либо cookie, либо cake

`$grep '<.*>' page.html` # - < появляется перед символом >

`$grep -F w. frost` # - отключить регулярные выражения

`$fgrep w. frost` # - поиск без регулярных выражений

Поиск по списку шаблонов из файла:

`$grep -f <filename> ...`

8 awk {print}

Печать второго слова в каждой строке:

```
$awk '{print $2}' /etc/hosts
```

```
$echo Efficient fun Linux | awk '{print $1 $3}' # - без пробела
```

```
$echo Efficient fun Linux | awk '{print $1, $3}' # - с пробелом
```

Выводить со второй строки:

```
$df / /data | awk 'FNR>1 {print $4}'
```

Любое количество двоеточий:

```
$echo efficient:::::linux | awk -F':' '{print $2}'
```

9 head, tail, tac

```
$head -n3 animals.txt # - первые три строки файла
```

```
$tail -n3 alphabet # - последние 3 строки
```

```
$tail -n+25 alphabet # - с 25-й строки файла
```

```
$head -n4 alphabet | tail -n1 # - только четвертую строку
```

```
$head -n8 alphabet | tail -n3 # - строки с шестой по восьмую
```

```
$head -4 alphabet # - = head -n4 alphabet
```

```
$tail -3 alphabet # - = tail -n3 alphabet
```

```
$tail +25 alphabet # - = tail -n+25 alphabet
```

```
$tac filename # - вывести строки в обратном порядке
```

10 paste

Объединить строки в столбцы, разделенные символом табуляции:

```
$paste title-words1 title-words2
```

```
$paste -d, title-words1 title-words2 # - разделитель запятая
```

```
$paste -d "\n" title-words1 title-words2 # - чередовать строки
```

Строки каждого файла соединяются в одну:

```
$paste -d, -s title-words1 title-words2
```

11 tr

Преобразование двоеточий в символы новой строки:

```
$echo $PATH | tr : "\n"
```

Перевод а в А, b в В и т. д.

```
$echo efficient | tr a-z A-Z
```

Преобразование пробелов в символы новой строки:

```
$ echo Efficient Linux | tr " " "\n"
```

Удаление пробелов и знаков табуляции

```
$ echo efficient linux | tr -d ' \t'
```

12 rev

Переворачивает символы задом наперед в каждой строке ввода:

```
$echo Efficient Linux! | rev
```

Вывести на экран последнее слово из каждой строки:

```
$rev celebrities | cut -d' ' -f1 | rev
```

13 Неразобранное

Полезные команды:

```
$cat /etc/os-release # - информация о версии системы
```

```
$netstat -tulpen # - Открытые порты
```

```
$curl -li https://localhost # - Содержимое сайта
```

```
env # - получить все системные переменные окружения
```

```
export <var>=<value> # - задать переменную окружения
```