Linux command line

Содержание

1 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!
```

2 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
```

3 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
```

4 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 {} " "
```

5 yes

```
$yes   # - Выводит «y» по умолчанию
$yes woof!   # - Повторять любую другую строку
```

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

7 less

Сочетания клавиш при просмотре с помощью команды less:

Сочетание	Действие
h	Справка
Пробел	Посмотреть следующую страницу
b	Посмотреть предыдущую страницу
Enter	Прокрутить вниз на одну строку
<	Перейти к началу документа
>	Перейти к концу документа
/	Поиск текста вперед (введите текст и нажмите Enter)
?	Поиск текста назад (введите текст и нажмите Enter)
n	Найти следующее вхождение искомого текста
q	Выйти

8 Вывод текста

8.1 head

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

8.2 tail

```
$tail -n3 alphabet   # - последние 3 строки
$tail -n+25 alphabet   # - с 25-й строки файла
$tail -3 alphabet   # - = tail -n3 alphabet
$tail +25 alphabet   # - = tail -n+25 alphabet
```

8.3 tac

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

8.4 fold

```
$fold -w40 title.txt   # - вывести текст с шириной не больше 40
```

8.5 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
```

9 Преобразование текста

9.1 tr

Преобразование двоеточий в символы новой строки:

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

Перевод a в A, b в B 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'
```

9.2 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
```

9.3 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
```

9.4 awk

Выполнить программу:

```
$awk program input-files
```

Выполнить несколько программ:

```
$awk -f program-file1 -f program-file2 -f program-file3 input-files
```

Слово `BEGIN` - действие перед обработкой ввода команды `awk`.
Слово `END` - действие после обработки ввода команды `awk`.

```
$awk 'FNR<=10' myfile   # - Выводит 10 строк и завершается
```

Поменять местами два слова:

```
$echo "linux efficient" | awk '{print $2, $1}'
```

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 4 колонку:

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

Любое количество двоеточий:

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

```
$awk '{print $NF}' celebrities   # - вывести последнее слово
$echo efficient linux | awk '/efficient/'   # - вхождения строки
awk: $3~/^[A-Z]/   # - начинается ли третье поле с заглавной буквы
```

Пример программы:

```
$awk -F'\t' \
   ' BEGIN {print "Recent books:"} \
   $3~/^201/{print "-", $4, "(" $3 ").", "\"" $2 "\""} \
   END {print "For more books, search the web"} ' \
   animals.txt
```

Просуммировать числа от 1 до 100:

```
$seq 1 100 | awk '{s+=$1} END {print s}'
```

Найти дубликать картинок:

```
$ md5sum *.jpg \
| awk 'counts[$1]++; names[$1]=names[$1] " " $2 \
END {for (key in counts) print counts[key] " " key ":" names[key]}'\
| grep -v '^1 ' \
| sort -nr
```

9.5 sed

Выполнить сценарий:

```
$sed script input-files
```

Выполнить несколько сценариев:

```
$sed -e script1 -e script2 -e script3 input-files
```

Выполнить несколько сценариев из файлов:

```
$sed -f script-file1 -f script-file2 -f script-file3 input-files
```

`$sed 10q myfile` # - выводит 10 строк и завершается
`$echo image.jpg | sed 's/jpg/.png/'` # - заменить .jpg на .png
`$sed 's/.* //' celebrities` # - вывести последнее слово

Можно использовать другие символы для разделения:

```
s_one_two_
```

Нечувствительный к регистру:

```
$echo Efficient Stuff | sed "s/stuff/linux/i"
```

Заменяет все вхождения «f»:

```
$echo efficient stuff | sed "s/f/F/g"
```

`$seq 10 14 | sed 4d` # - удаляет четвертую строку
Удаляет строки, заканчивающиеся на нечетные цифры:

```
$seq 101 200 | sed '/[13579]$/d'
```

Использование ссылок на подвыражения \1, \2, ... :

```
$ls | sed "s/image\.jpg\.\([1-3]\)/image\1.jpg/"
```

10 11 способов запуска команды

10.1 Условные списки

```
$cd dir && touch new.txt   # - запускаются до первой ошибки
$cd dir || mkdir dir   # - запускаются до первого успеха
```

10.2 Безусловные списки

Последовательный запуск команд:

```
$sleep 300; echo "remember to walk the dog" | mail -s reminder
$USER
```

10.3 Подстановка команд

```
$mv $(grep -l "Artist: Kansas"*.txt) kansas
$echo Today is $(date +%A).
$echo Today is $(echo $(date + %A) | tr a-z A-Z )!
```

10.4 Подстановка процесса

```
$cat <(ls -1 | sort -n)
$cp <(ls -1 | sort -n) /tmp/listing
$diff <( ls *.jpg | sort -n ) <( seq 1 1000 | sed 's/$/.jpg/' )
```

10.5 Передача команды в bash в качестве аргумента

```
$bash -c "ls -l"
$sudo bash -c 'echo "New log file" > /var/log/custom.log'
```

10.6 Передача команды в bash через стандартный ввод

```
$echo "ls -l" | bash
```

10.7 Удаленное выполнение однострочника с помощью ssh

Создает outfile на локальном хосте:

```
$ssh myhost.example.com ls > outfile
```

Создает outfile на удаленном хосте:

```
$ssh myhost.example.com "ls > outfile"
```

С параметром -T, чтобы удаленный ssh-сервер не выделял терминал:

```
$echo "ls > outfile ssh -T myhost.example
```

10.8 Запуск списка команд с помощью xargs

У команды find использовать -print0 вместо -print, то строки будут разделяться нулевым символом. xargs -0 - чтобы разделителем служил нулевой символ.

```
$find . -type f -name \*.py -print0 | xargs -0 wc -l

$ls | xargs echo   # - уместить как можно больше входных строк
$ls | xargs -n1 echo   # - один аргумент в каждой команде echo
$ls | xargs -n3 echo   # - три аргумента в каждой команде echo
```

Параметр -I определяет место входных строк в сгенерированной команде. XYZ в качестве прототипа:

```
$ls | xargs -I XYZ echo XYZ is my favorite food

$find . -maxdepth 1 -name \*.txt -type f -print0 \
   | xargs -0 rm
```

10.9 Фоновое выполнение команды

Подсчет символов в огромном файле:

```
$wc -c my_extremely_huge_file.txt &
```

Все три команды выполняются фоном:

```
$command1 & command2 & command3 &
```

Команды управления заданиями:

Команда	Значение
bg	Переместить текущее приостановленное задание в фоновый режим
bg %n	Переместить приостановленное задание номер n в фоновый режим (пример: bg %1)
fg	Переместить текущее фоновое задание на передний план
fg %n	Переместить фоновое задание номер n на передний план (пример: fg %2)
kill %n	Завершить фоновое задание номер n (пример: kill %3)
jobs	Просмотр списка заданий оболочки

Перевести команду в фон: нажать `Ctrl-Z`, а затем `bg`.

10.10 Явные подоболочки

```
$(cd /usr/local && ls)
$ cat package.tar.gz | \
   (mkdir -p /tmp/other && cd /tmp/other && tar xzvf -)
```

Копировать файлы в существующий каталог на другом хосте через SSH:

```
$tar czf - dir1 | ssh myhost '(cd /tmp/dir2 && tar xvf -)'
```

10.11 Замена процесса

ls заменяет дочернюю оболочку, запускается и завершает работу:

```
$exec ls
```

11 Создание дерзких однострочников

Сдвинуть нумерацию файлов:

```
$ paste <(echo 1..10.jpg | sed 's/ /\n/g') \
        <(echo 0..9.jpg | sed 's/ /\n/g') \
   | sed 's/^/mv /' \
   | bash
```

Найти 17-й символ:

```
$echo {A..Z} | awk '{print $(17)}'
```

```
$echo {A..Z} | sed 's/ //g' | cut -c17
```

Вывести названия месяцев:
```
$echo 2021-{01..12}-01 | xargs -n1 date +%B -d
```

Количество символов в самом длинном имени файла:
```
$ls | awk '{print "echo -n", $0, "| wc -c"}' | bash \
      | sort -nr | head -n1
```

Проверка совпадающих пар файлов:
```
$diff <(ls *.jpg | sed 's/\.[^.]*$//') \
      <(ls *.txt | sed 's/\.[^.]*$//') \
   | grep '^[<>]' \
   | awk '/</{print $2 ".jpg"} />/{print $2 ".txt"}'
```
Другой способ:
```
$ls -1 $( ls *.{jpg,txt} \
   | sed 's/\.[^.]*$//' \
   | uniq -c \
   | awk '/^ *1 /{print $2 "*"}' )
```

Создание 1000 файлов со случайными именами и случайными словами:
```
$ yes 'shuf -n $RANDOM -o $(pwgen -N1 10).txt /usr/share/dict/words'
\
      | head -n 1000 \
      | bash
```

12 Управление буфером обмена

Копирование и вставка из первичного буфера:
```
$echo -n | xclip
$xclip -o
```

Копирование и вставка из системного буфера:
```
$echo https://oreilly.com | xclip -selection clipboard
$xclip -selection clipboard -o
```

13 Переменные окружения

Получить все системные переменные окружения:

```
   $printenv | sort -i | less
   $env
$MY_VARIABLE=10   # - установить значение локальной переменной
$export E="variable"   # - установить переменную оболочки
```

14 web

```
$wget
$curl -li https://localhost   # - содержимое сайта
```

Парсинг страницы html по классам объектов:

```
   $curl -s https://efficientlinux.com/areacodes.html \
      | hxnormalize -x \
      | hxselect -c -s@ '#ac .ac, #ac .state, #ac .cities'

$lynx -dump https://efficientlinux.com/areacodes.htm
```

15 Неразобранное

Полезные команды:

```
   $cat /etc/os-release   # - информация о версии системы
   $netstat -tulpen   # - Открытые порты
   $ps -uax   # - все запущенные процессы для всех пользователей:
```

Код возврата:

```
   $?
```

Обработка файла построчно:

```
   $cat /etc/hosts | while read line; do
      echo "$line wc -c
   done
```