Linux

Работа с файлами

Содержание

Команды работы с файлами:

- echo
- cat
- more
- less
- joe *
- nano*
- grep
- find
- sed
- cmp
- diff
- patch

- touch
- cp
- mv
- rm *
- dd
- head
- tail
- tee
- cut
- tr

- echo(bash)
- basename
- dirname
- wc
- uniq
- sort
- yes *

echo
ЗАПИСАТЬ АРГУМЕНТЫ В СТАНДАРТНЫЙ ВЫВОД

echo [OPTION]... [STRING]...

Выводит строку string на стандартный вывод (stdout). В конце выполняется перевод каретки

-n Не печатать завершающий символ новой строки

-e Обрабатывать esc-последовательности

\f	перевод страницы
\n	перевод строки
\r	возврат каретки
\t	горизонтальная табуляция
\v	вертикальная табуляция

$ echo “Hello, world”
$ echo –n “Hello, ”; echo world

```
20:48:00 MR24.6 porta-one@etsys.intra:/dev
> echo "hello world!"
-bash: !": event not found
20:48:16 MR24.6 porta-one@etsys.intra:/dev
> echo 'hello world!'
hello world!
```

cat

ОБЪЕДИНИТЬ И НАПЕЧАТАТЬ ФАЙЛЫ

cat [OPTION] [FILE]...

Утилита cat последовательно читает файлы и пишет их в стандартный вывод. Аргументы file обрабатываются в порядке их следования в командной строке. Если file задан как дефис или вовсе отсутствует, то cat производит чтение со стандартного ввода

-b	Нумеровать непустые выводимые строки, начиная с 1
-n	Нумеровать все выводимые строки, начиная с 1
-v	Выводить непечатаемые символы в читабельном виде

```
$ cat file1 file2 > file3
$ cat file1 - file2 - file3
```

```
$ cat > newfile
```

```
> cat 1.xtx
#!/bin/bash
Parameter=$1
File=`cat 1.txt| grep -rE "\[$Parameter\]" -A 1`
echo "$File"
```

less
ОТОБРАЖЕНИЕ СОДЕРЖИМОГО ФАЙЛА

less [options] files ...

Быстрое и гибкое отображение, перемещение, поиск в больших файлах. Команды управления основаны на командах vi

-i, --ignore-case	Регистронезависимый поиск только для строчных букв
-I, --IGNORE-CASE	Регистронезависимый поиск
-J	Отобразить столбец состояния поиска (слева)
-n, --line-numbers	Не нумеровать строки
-N, --LINE-NUMBERS	Нумеровать строки
-S, --chop-long-lines	«Отсечение» длинных строк (без переноса)

```
$ less –SR big_file
```

Команды управления в less

Q :q Q :Q ZZ	Выход
e ^E j ^N CR DownArrow	Вперед на одну строку
y ^Y k ^K ^P UpArrow	Назад на одну строку
b ^B	Назад на окно
Z Space	Вперед на окно
d ^D	Вперед на половину окна
u ^U	Назад на половину окна
RightArrow	Вправо на половину окна
LeftArrow	Влево на половину окна
F	Вперед с ожиданием (эквивалентно «tail -f»)
g <	В начало файла
G >	В конец файла
:e [file]	Открыть файл file
:n	Перейти к следующему файлу

Команды управления в less

:p	Перейти к предыдущему файлу
:d	Удалить файл из списка
= ^G :f	Отобразить информацию о файле
/pattern	Прямой поиск по шаблону pattern
?pattern	Обратный поиск по шаблону pattern
n	Следующее вхождение (при поиске)
N	Предыдущее вхождение (при поиске)

```
find –maxdepth 1 –type f | xargs less
```

grep
ПОИСК ПО ШАБЛОНУ В ФАЙЛЕ

grep [options] PATTERN [FILE...]

Поиск совпадений по шаблону в указанных файлах или стандартном вводе, и отображение (строк)

-A NUM, --after-context=NUM	Отобразить также NUM строк «после» совпадения
-B NUM, --before-context=NUM	Отобразить также NUM строк «до» совпадения
-C NUM, --context=NUM	Отображать также NUM строк «до» и «после» совпадения
--colour, --color	Подсвечивать совпадения
-c, --count	Отобразить только число совпадений (строк)
-f FILE, --file=FILE	Считать шаблон из файла
-H, --with-filename	Отображать также имена файлов (не только совпадения)
-h, --no-filename	Не отображать имена файлов (только совпадения)
-i, --ignore-case	Регистронезависимый поиск
-L, --files-without-match	Отображать только имена файлов без совпадений

grep
ПОИСК ПО ШАБЛОНУ В ФАЙЛЕ

-l, --files-with-matches	Отображать только имена файлов с совпадениями
-n, --line-number	Отображать также и номера строк
-o, --only-matching	Отображать только совпадения (не всю строку)
-R, -r, --recursive	Обрабатывать файлы рекурсивно
-v, --invert-match	Инверсия совпадений
-w, --word-regexp	Только целые слова

```
> sudo less anaconda.log | grep -nE "Saving module i"
6:14:34:08,862 DEBUG    : Saving module iscsi_ibft
7:14:34:08,862 DEBUG    : Saving module iscsi_boot_sysfs
13:14:34:08,862 DEBUG    : Saving module ib_ipoib
14:14:34:08,862 DEBUG    : Saving module ib_cm
15:14:34:08,862 DEBUG    : Saving module ib_sa
16:14:34:08,862 DEBUG    : Saving module ib_mad
17:14:34:08,862 DEBUG    : Saving module ib_core
18:14:34:08,862 DEBUG    : Saving module ipv6
19:14:34:08,862 DEBUG    : Saving module iscsi_tcp
```

Регулярные выражения (REGEXP)

Регулярные выражения (regular expressions) – это система синтаксического разбора текстовых фрагментов по формализованному шаблону, основанная на системе записи образцов для поиска

Регулярное выражение, шаблон, маска (regular expression, pattern) – это шаблон (образец), задающий правило поиска

```
(\b([[:digit:]]{1,3}\b\.){3}(\b[[:digit:]]{1,3})\b)
```

```
(\b([01]{0,1}[[:digit:]]{1,2}|2[0-4][[:digit:]]|25[0-5])\b\.){3}
(\b([01]{0,1}[[:digit:]]{1,2}|2[0-4][[:digit:]]|25[0-5])\b)
```

Синтаксис REGEXP

- **Обычные символы (текст)**
- **Метасимволы (специальные символы)**
  - **[ ] \ ^ $. | ? * + ( ) { }**
  - Нуждаются в экранировании при использовании в качестве текста:
    - Одиночное – **«\»**
    - Групповое – между **«\Q» и «\E»**
- **Любой символ – «.»**
- **Наборы символов (один из перечисленных символов)**
  - Заключаются в **«[»**, **«]»**: **[abcABC]**
  - Диапазон задается через дефис **«-»**: **[a-cA-C]**
  - Инверсия (не входит) – **«^»**: **[^0-9]**
  - Именованные классы наборов:

    [:alnum:] (**«\w»**, инверсия – **«\W»**), **[:alpha:]**, **[:digit:]**, **[:lower:]**, **[:punct:]**, **[:space:]**, **[:upper:]**

Синтаксис REGEXP

- Позиция в строке
  - Начало строки – **«^»**, конец строки – **«$»**
  - Граница слова – **«\b»**, не граница слова – **«\B»**
- Последовательности
  - Равно **n** – **«{n}»**
  - От **n** до **m** (оба включительно) – **«{n, m}»**
  - Не менее **n** – **«{n,}»**
  - Не более **m** – **«{,m}»**
  - Ноль или более – **«*»**
  - Один или более – **«+»**
  - Ноль или один – **«?»**
- Перечисление
  - Наборы заключенные в **«(»** и **«)»**, и разделенные **«|»**: **(a|b)**

grep

-E Расширенное выражение REGEXP

GREP_OPTIONS Аргументы командной строки

```
$ grep -woE “[[:alnum:]]+” text | sort | uniq > dictionary
$ grep --color -rolE "192\.168\.[[:digit:]]{1,3}\.1" * | sort | uniq
$ grep -vnE --color “(^[[:space:]]*#|^[[:space:]]*$)” config
```

```
> grep 'Jan [[:digit:]]{2}.([[:digit:]]{2}:){2}[[:digit:]]{2}' -roE ./
grep: ./.lesshst: Permission denied
./CUS/Entries:Jan 16 16:03:11
./CUS/Entries:Jan 23 19:50:12
grep: ./.asterisk_history: Permission denied
```

```
> sudo less anaconda.log | grep -nE "([[:digit:]]{1,3}\:){3}"
126:14:47:18,824 DEBUG    : notifying kernel of 'change' event on device /sys/dev
ices/pci0000:00/0000:00:01.1/host1/target1:0:0/1:0:0:0/block/sr0
07:32:24 MR24.6 porta-one@etsys.intra:/var/log
>
```

find
ПОИСК ФАЙЛОВ

find путь ... выражение

Утилита find рекурсивно спускается по дереву каталогов каждого пути, указанного аргументами путь, вычисляя выражение (критерий) для каждого файла в дереве

-empty	**Если текущий файл или каталог пусты**
-exec prg [arg ...] [{}] \;	**Запуск программы prg для каждого подходящего элемента директории**
-group grp_name	**Если файл принадлежит группе grp_name**
-iname pattern	**Подобен -name, но сравнение не учитывает регистр**
-inum num	**Если номер индексного дескриптора файла равен num**
-maxdepth n	**Заставляет спускаться не более чем на n уровней каталогов ниже аргументов командной строки (директорий)**
-mindepth n	**Не применять любые тесты или действия на уровнях меньше n**
-name pattern	**Если последний компонент пути файла (его имя) подпадает под шаблон**
-print	**Вывод путь текущего файла**
-user user	**Если файл принадлежит пользователю с именем user**
-size n[ckMGTP]	**Если размер файла в 512-байтных блоках, при округлении вверх, равен n (или согласно модификаторов ckMGTP)**

find

-newer file	Файл имеет более свежее время модификации чем file
-type type	Если типом текущего файла является type
b	Блочный специальный
c	Символьный специальный
d	Каталог
f	Обычный файл
l	Символическая ссылка
p	Именованный канал (FIFO)
s	Сокет
-B **	Ссылка на время создания индексного дескриптора
-a **	Ссылка на время последнего доступа к файлу
-c **	Ссылка на время последней модификации статуса файла
-m **	Ссылка на время последней модификации файла
**min n	Разница в минутах (с текущим временем)
newer file	Новее указанного файла
time [-+]n[smhdw]	Заданная разница с текущим временем

```
$ find . -name “*.c” -print
$ find . -newer ref_file -user root –print
$ find . -type f -exec echo {} \;
```

sed

ПОТОКОВЫЙ РЕДАКТОР

sed {[-e cmd]|cmd} [-f cmd_file] [-i extension] [file ...]

Считывает указанные файлы или стандартный ввод, модифицирует их согласно указанных команд и выводит на стандартный вывод. Если указана опция -i – модифицирует непосредственно сам файл. Множественные команды могут быть заданы при помощи -e и -f

-E Интерпретировать REGEXP как расширенное

-e command Дополнение списка команд

-f cmd_file Дополнение списка команд (из файла)

-i extension Модификация непосредственно самого файла, резервная копия с указанным расширением

```
$ sed -e ‘s/oldstuff/newstuff/g’ inputFileName > outputFileName
$ sed –i.bak ‘s/abc/def/’ file
```

cmp

СРАВНИТЬ ДВА ФАЙЛА

cmp файл1 файл2 [пропуск1 [пропуск2]]

Утилита cmp сравнивает два файла файл1 и файл2 любого типа и пишет результат в стандартный вывод. По умолчанию cmp не выдаёт никаких сообщений, если файлы одинаковы; если же они различаются, сообщается номер байта и строки, где обнаружено первое различие

пропуск1 Пропустить указанное число строк в первом файле (до сравнения)

пропуск2 Пропустить указанное число строк во втором файле (до сравнения)

```
$ cmp file1 file2
$ cmp file1 file2 1 2
```

```
> sudo cmp cron.2.bz2 yum.log.1.bz2 6 23
cron.2.bz2 yum.log.1.bz2 differ: byte 1, line 1
07:52:16 MR24.6 porta-one@etsys.intra:/var/log
> sudo cmp cron.2.bz2 yum.log.1.bz2
cron.2.bz2 yum.log.1.bz2 differ: byte 11, line 1
```

diff

СРАВНИТЬ ФАЙЛЫ ПОСТРОЧНО

diff [OPTION]... FILES

-i, --ignore-case	Игнорировать разницу в регистре букв
-b	Игнорировать разницу в числе пробелов между словами
-B	Игнорировать разницу в пустых строках
-c, -C NUM, --context[=NUM]	Контекстный формат, число строк в контексте
-u, -U NUM, --unified[=NUM]	Унифицированный формат, число строк в контексте
-r, --recursive	Выполнять рекурсивное сравнение
-N, --new-file	Считать отсутствующие файлы пустыми
-x PAT, --exclude=PAT	Исключить файлы по шаблону PAT

```
$ diff –u my.old my.new > my.patch
$ diff –urBNE proj_dir.orig proj_dir > proj.patch
```

patch
НАЛОЖИТЬ ПАТЧ

patch [options] [origfile [patchfile]] [+ [options] [origfile]]...

patch [options] < patchfile

Утилита накладывает патч(заплатку), созданный утилитой diff (одного из 4-х видов), на оригинал, создавая новую версию, при этом может создавать резервные копии

-b Сохранять резервные копии

--dry-run Проверить патч, но не накладывать

-d dir, --directory=dir Предварительно перейти в указанную директорию

-E, --remove-empty-files После наложения патчей удалить пустые файлы

-i patchfile Считать патч из указанного файла вместо stdin

-p[number], --strip[=number] Задать число элементов пути («/»), которое будет удалено в именах файлов внутри патча. По умолчанию: для полного пути – только имя файла, для относительного – весь путь

```
$ patch < my.patch
$ patch –Cd x-proj < x-my.patch
```

cp
КОПИРОВАТЬ ФАЙЛЫ

cp [-R] [-f | -i | -n] [-lpv] исходный_файл целевой_файл

cp [-R] [-f | -i | -n] [-lpv] исходный_файл ... целевой_каталог

Жесткие и символические ссылки копируются как объекты, на которые они указывают (а не специальные файлы). Целевой каталог должен существовать (если не указана опция -R).

Для копирования иерархии правильнее использовать tar или cpio!

- **-f** Принудительная замена файлов (по возможности)
- **-i** С запросом подтверждения замены файлов
- **-p** Сохранять атрибуты исходных файлов при замене (по возможности)
- **-R** Рекурсивно (вместе со всем деревом иерархии)

```
$ cp file1 file2 file3 dir
$ cp –vR dir /tmp
$ cp –fp file1 file2
```

mv

ПЕРЕМЕСТИТЬ ФАЙЛЫ

mv [-f | -i] [-v] источник цель

mv [-f | -i] [-v] источник ... каталог

- **Утилита mv переименовывает файл, заданный аргументом источник, в целевой путь, заданный аргументом цель (такая форма подразумевается, когда последний операнд не является именем уже существующего каталога).**
- **Утилита mv перемещает каждый файл источник в целевой файл в существующем каталоге, заданным операндом каталог.**

-f	Не запрашивать подтверждение перед перезаписью целевого пути
-i	Запрос подтверждения на перезапись

```
$ mv file1 file2
$ mv –vf dir1 dir2
$ mv * /home/user/new_dir
```

head

ВЫВЕСТИ ПЕРВЫЕ СТРОКИ ФАЙЛА

head [-n число | -c байт] [файл ...]

Этот фильтр выводит первые число строк или байт каждого из указанных файлов либо стандартного ввода, если файлы не были указаны. Если аргумент число опущен, по умолчанию он принимается равным 10

$ head –n 5 file

```
> sudo head -n 5 anaconda.log
14:34:05,574 INFO    : kernel command line: initrd=initrd.img BOOT_IMAGE=vmlinuz


14:34:05,574 DEBUG   : readNetInfo /tmp/s390net not found, early return
14:34:05,574 INFO    : anaconda version 13.21.117 on x86_64 starting
14:34:05,643 INFO    : 2055528 kB are available
```

tail
ВЫВЕСТИ ПОСЛЕДНЮЮ ЧАСТЬ ФАЙЛА

tail [-F | -f | -r] [-q] [-b номер | -c номер | -n номер] [файл ...]

Утилита tail выводит содержимое файла файл или, по умолчанию, своего стандартного ввода, на стандартный вывод. Вывод начинается с определённого байта или строки входного файла.

Числа, перед которыми стоит знак плюс указывают позицию относительно начала входного файла. Числа, перед которыми стоит знак минус (или знак отсутствует), указывают позицию относительно конца входного файла.

-c номер Вывод начнётся с байта номер

-f Не выходить при достижении конца (ждать новых данных)

-F Аналогично -f, плюс проверка не был ли файл переименован (привязка к имени)

-n номер Вывод начнется со строки номер

-q Подавляет печать заголовков в случае, когда одновременно просматриваются несколько файлов

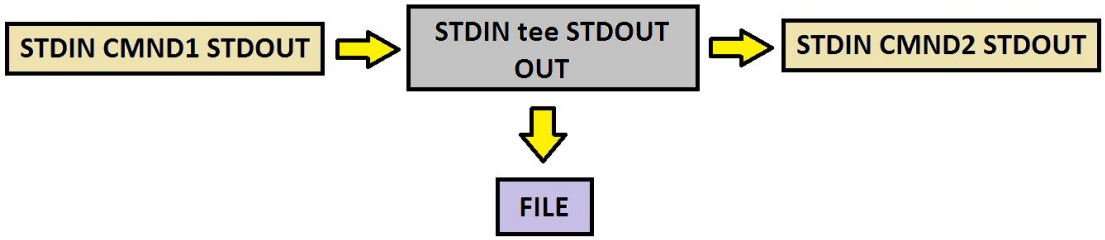
```
$ tail –Ff –n 0 /var/log/messages
$ tail –n +20 file | tail –n 10
```

tee
ДУБЛИРОВАТЬ СТАНДАРТНЫЙ ВВОД

tee [-a] [файл ...]

Утилита tee копирует стандартный ввод в стандартный вывод, помещая копию в ноль или более файлов

-a Добавлять вывод к файлам, а не перезаписывать их



$ tail -Ff /var/log/messages | grep --line-buffered kernel | tee -a LOG

```
> ls
echo.log
12:42:24 MR24.6 root@etsys.intra:/home/test
> echo 'Hello world!'
Hello world!
12:42:40 MR24.6 root@etsys.intra:/home/test
> echo 'Hello world!'| tee echo.log
Hello world!
12:42:58 MR24.6 root@etsys.intra:/home/test
> cat echo.log
Hello world!
```

touch

ИЗМЕНИТЬ ВРЕМЯ ДОСТУПА И МОДИФИКАЦИИ ФАЙЛА, СОЗДАТЬ ФАЙЛ

touch [OPTION]... FILE...

Установка времени модификации и доступа к файлам, создание файлов

- **-c** Не создавать файл
- **-a** Изменить время доступа к файлу
- **-m** Изменить время модификации файла
- **-r file** Сослаться на данные файла file при установке время доступа и модификации
- **-t** Установить время согласно модификатора [[CC]YY]MMDDhhmm[.SS]]

```
$ touch file
$ touch –r ref_file file
$ touch -d '2007-01-31 8:46:26' file
```

dd
КОНВЕРТИРОВАТЬ И КОПИРОВАТЬ ФАЙЛ

dd [operands ...]

Утилита dd копирует стандартный ввод на стандартный вывод блоками по 512 байт

bs=n	Размер блока (block size)
count=n	Копировать только n блоков
ibs=n	Размер входного блока (input block size)
if=file	Входной файл (input file)
iseek=n	Пропустить во входном файле n блоков (skip)
obs=n	Размер выходного блока (output block size)
of=file	Выходной файл (output file)
oseek=n	Пропустить в выходном файле n блоков (seek)
seek=n	Пропустить в выходном файле n блоков
skip=n	Пропустить n блоков от начала входного файла

```
# dd if=/dev/ad0 of=/dev/null bs=1M
# dd if=boot.flp of=/dev/fd0
```

tr

ПРЕОБРАЗОВАТЬ СИМВОЛЫ

tr [OPTION]... SET1 [SET2]

Преобразование, сжатие и/или удаление символов со стандартного ввода в стандартный вывод. Символы string1 преобразуются в символы string2 (согласно позиций). Если 1-я строка длиннее, то последний символ string2 дублируется пока не закончится string1.

-d Удалить символы встречающиеся в string1

```
$ echo a black cat | tr -d a
blck ct
```

-s Удалить повторяющиеся символы из string1 (сжать)

```
$ echo many      blank     spaces | tr -s ' '
many blank spaces
```

-c Заставляет команду работать с символами, которые отсутствуют в наборе 1: символы, перечисленные в наборе1 не используются в работе, а все остальные - используются.

```
$ echo a black caty| tr -cd b-[:cntrl:][:blank:]
blck cty
```

```
$ tr -cs "[:alpha:]" "\n" < file1
$ tr "[:lower:]" "[:upper:]" < file1
$ tr -cd "[:print:]" < file1
```

```
$ tr -cs a-zA-Z '\n' < /etc/fstab

dev
hda
...
Превращает файл /etc/fstab/ в список слов этого файла.
```

cut ВЫРЕЗАТЬ ОПРЕДЕЛЁННЫЕ ЧАСТИ ИЗ КАЖДОЙ СТРОКИ ФАЙЛА

cut -b список [файл ...]

cut -c список [файл ...]

cut -f список [-d разделитель] [-s] [файл ...]

Утилита cut вырезает указанные аргументами список части из каждой строки каждого файла и пишет их в стандартный вывод

-b список Аргумент список задаёт позиции в байтах

-c список Аргумент список задаёт позиции в символах

-d разделитель Использовать указанный аргументом разделитель символ как разделитель полей вместо символа табуляции

-f список Поля, разделённые символом разделителя полей

-s Пропускать строки, в которых не встречается символ разделителя. Если эта опции не указана, такие строки выводятся в неизменённом виде

$ ls -la | tail -n +2 | tr -s ' ' | cut -s -d ' ' -f 3 | sort | uniq

```
> cat echo.log
Column1 ^! Column2 ^! Column3 ^! Column4
------------------------
------------------------
Column1 ^! Column2 ^! Column3
13:01:50 MR24.6 root@etsys.intra:/home/test
> cut -s -d '^' -f 1,2 echo.log
Column1 ^! Column2
Column1 ^! Column2
```

basename, dirname

ВЕРНУТЬ ФАЙЛОВУЮ ИЛИ КАТАЛОГОВУЮ ЧАСТЬ ПУТИ

basename NAME [SUFFIX]

dirname NAME

Утилита basename удаляет из строки name любой префикс (часть пути), оканчивающийся последней косой чертой «/» в строке, предварительно удалив косые черты в конце строки (фактически, остается только имя файла).

Утилита dirname удаляет файловую часть, начиная с последней «/» до конца строки строка, предварительно удалив «/» из конца строки, и пишет результат в стандартный вывод.

```
$ basename /etc/rc.conf
$ basename /etc/rc.conf .conf
$ dirname /etc/rc.conf
```

```
> basename /home/test/echo.log
echo.log
13:24:06 MR24.6 root@etsys.intra:/home/test
> dirname /home/test/echo.log
/home/test
```

wc ПОДСЧЁТ КОЛИЧЕСТВА СЛОВ, СТРОК, СИМВОЛОВ И БАЙТОВ

wc [-clmw] [файл ...]

Утилита wc пишет в стандартный вывод число строк, слов и байтов, содержащихся в каждом входном файле, заданном аргументом файл, или прочитанных из стандартного ввода.

Порядок вывода всегда имеет следующий формат: строки, слова, байты и имя файла. Действие команды по умолчанию равносильно указанию опций -c, -l и -w.

-c	Число байтов, содержащихся в каждом входном файле
-l	Число строк, содержащихся в каждом входном файле
-m	Число символов, содержащихся в каждом входном файле
-w	Число слов, содержащихся в каждом входном файле

```
$ grep –w WORD file | wc –l
$ wc text.db
```

```
> grep -roE Column. echo.log
Column1
Column2
Column3
Column4
Column1
Column2
Column3
13:30:21 MR24.6 root@etsys.intra:/home/test
> grep -roE Column. echo.log| wc -clmw
      7       7      56      56
```

uniq

ВЫВЕСТИ ИЛИ ОТФИЛЬТРОВАТЬ ПОВТОРЯЮЩИЕСЯ СТРОКИ В ФАЙЛЕ

uniq [OPTION]... [INPUT [OUTPUT]]

Утилита uniq читает вход_файл, сравнивает соседние строки и пишет копию каждой уникальной входной строки в вых. файл. Вторая и последующие копии повторяющихся соседних строк не записываются. Повторяющиеся входные строки не распознаются, если они не следуют строго друг за другом, поэтому может потребоваться предварительная сортировка файлов

- **-c** Перед каждой строкой выводить число повторений этой строки на входе
- **-d** Выводить только те строки, которые повторяются
- **-f N_полей** Игнорировать при сравнении первые N_полей полей (слов*) каждой строки
- **-s N_символов** Игнорировать при сравнении первые N_символов символов каждой строки
- **-u** Выводить только те строки, которые не повторяются на входе
- **-i** Сравнивать строки без учёта регистра

```
$ uniq file.in file.out
$ uniq –iu file.in > file.original
```

```
> grep -roE Column. echo.log| uniq
Column1
Column2
Column3
Column4
Column1
Column2
Column3
```

```
> grep -roE Column. echo.log| sort| uniq -c
      2 Column1
      2 Column2
      2 Column3
      1 Column4
```

sort

СОРТИРОВАТЬ СТРОКИ ТЕКСТОВЫХ ФАЙЛОВ

sort [OPTION]... [FILE]...

-f, --ignore-case	Игнорировать регистр символов
-M, --month-sort	Сортировать как названия месяцев
-n, --numeric-sort	Сортировать как числа
-r, --reverse	Отобразить в обратном порядке
-k, --key=POS1[,POS2]	Сортировать по ключу, перечню полей (столбцов)
-t, --field-separator=SEP	Использовать SEP как разделитель полей (столбцов)
-u, --unique	Выводить только уникальные записи

```
$ ls -la | tail -n +2 | sort -k 2 –n
$ grep -woE “[[:alnum:]]+” text | sort | uniq > dictionary
```

```
> grep -roE Column. echo.log| sort -ur
Column4
Column3
Column2
Column1
```

YES

yes [STRING]

yes — UNIX-команда, бесконечно выводящая аргументы командной строки, разделённые пробелами до тех пор, пока не будет убита (например, командой kill). Если в командной строке не задано аргументов, то бесконечно выводит строку «y».

Позволяет получить загрузку CPU ~100%

Создать файл Bigfile, содержащий 5000 строк “I want to make dig file”:

```
$ yes I want to make big file! | head -5000 > Bigfile
```

```
yes | rm -i *.txt
```

softlink

СИМВОЛЬНЫЕ ССЫЛКИ

Символьная ссылка (также симлинк от англ. Symbolic link, символическая ссылка) — специальный файл в файловой системе, для которого не формируются никакие данные, кроме одной текстовой строки с указателем. Эта строка трактуется как путь к файлу, который должен быть открыт при попытке обратиться к данной ссылке (файлу). Символьная ссылка занимает ровно столько места в файловой системе, сколько требуется для записи её содержимого (нормальный файл занимает как минимум один блок раздела).

- **Целью ссылки может быть любой объект — например, другая ссылка, файл, папка, или даже несуществующий файл (в последнем случае при попытке открыть его должно выдаваться сообщение об отсутствии файла).** Ссылка, указывающая на несуществующий файл, называется висячей.

- **Практически символьные ссылки используются для более удобной организации структуры файлов на компьютере,** так как позволяют одному файлу или каталогу иметь несколько имён, различных атрибутов и свободны от некоторых ограничений, присущих жёстким ссылкам (последние действуют только в пределах одного раздела и не могут ссылаться на каталоги).

Создание символьной ссылки:

$ ln -s файл имя_ссылки

```
> ls -lah
total 16K
drwxr-xr-x 2 root root 4.0K Jan 23 14:44 .
drwxr-xr-x 3 root root 4.0K Jan 23 14:37 ..
lrwxrwxrwx 1 root root   19 Jan 23 14:38 echo_s.log -> /home/test/echo.log
```

hardlink

ЖЕСТКИЕ ССЫЛКИ

Жёсткой ссылкой (англ. hard link) в UFS-совместимых файловых системах называется структурная составляющая файла — описывающий его элемент каталога.

•Файл в UFS представляет собой структуру блоков данных на диске, имеющую уникальный индексный дескриптор (или i-node) и набор атрибутов (метаинформацию). Жёсткая ссылка связывает индексный дескриптор файла с каталогом и дает ему имя. **У файла может быть несколько жёстких ссылок: в таком случае он будет фигурировать на диске одновременно в различных каталогах и/или под различными именами.**

•Количество жёстких ссылок файла сохраняется на уровне файловой системы в метаинформации. Файлы с нулевым количеством ссылок перестают существовать для системы и, со временем, будут перезаписаны физически. В файловых системах unix-подобных ОС и NTFS при создании файла на него автоматически создаётся одна жёсткая ссылка (на то место файловой системы, в котором файл создаётся). Дополнительную ссылку можно создать с помощью команды ln. Все ссылки одного файла равноправны и неотличимы друг от друга — нельзя сказать, что файл существует в таком-то каталоге, а в других местах есть лишь их копии. **Удаление любой из ссылок приводит к удалению файла лишь в том случае, когда удалены все остальные жёсткие ссылки на него.**

•Большинство программ не различают жёсткие ссылки одного файла, даже системный вызов для удаления файла в UNIX называется unlink (англ.)русск., так как он предназначен для удаления жёсткой ссылки файла.

•В связи с тем, что жёсткие ссылки ссылаются на индексный дескриптор, уникальный в пределах дискового раздела, создание жёсткой ссылки на файл в каталоге другого раздела невозможно. Для преодоления этого ограничения используются символьные ссылки.

Создание жесткой ссылки:

```
$ ln файл имя_ссылки
```

СПАСИБО ЗА ВНИМАНИЕ!