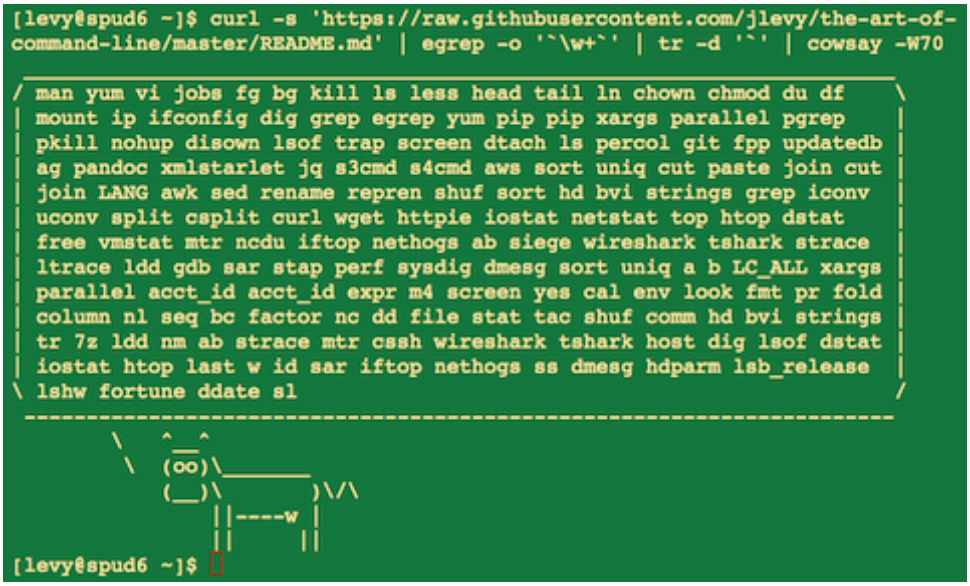
Искусство командной строки

- Описание
- Основы
- Ежедневное использование
- Обработка файлов и информации
- Отладка системы
- В одну строчку
- Сложно, но полезно
- OS X only
- Windows only
- Больше информации по теме
- Дисклеймер



Продвинутому использованию командной строки зачастую не уделяют достаточного внимания. О терминале говорят как о чем-то мистическом. На самом же деле, это умение очевидно (и не очевидно) увеличивает Вашу продуктивность в работе. Данный документ является подборкой заметок и советов, которые я нашел для себя полезными, работая с командной строкой в Linux. Некоторые из них – простые и очевидные, но некоторые - довольно сложные. И предназначены для решения конкретных задач. Это небольшая публикация, но если Вы уже знаете обо всем, что тут написано, и можете вспомнить, как это все использовать – вы много знаете!

Этот гайд является результатом работы большого числа авторов и переводчиков Многое из того, что тут написано, изначально появилось на Quora, начав идею там, похоже, что стоит развить ее на Github, где обитают люди, которые талантливее меня и могут предлагать улучшения данной подборки. Если Вы заметили ошибки (во всех вариантах перевода), пожалуйста **оставьте тикет или добавьте пулл-реквест** (заранее изучив описание и посмотрев на уже созданнные тикеты и пулл-реквесты).

Описание

Основное:

- Данная публикация предназначена как для новичков, так и для опытных людей. Цели: *объемность* (собрать все важные аспекты использования командной строки), *практичность* (давать конкретные примеры для самых частых юзкейсов) и *краткость* (не стоит углубляться в неочевидные вещи, о которых можно почитать в другом месте).
- Этот документ написан для пользователей Linux, за исключением секций "OS X only" и "Windows only". Все остальное подходит и может быть установлено под все UNIX/OS X системы (и даже Cygwin).
- Фокусируемся на интерактивном Баше, но многие вещи также могут быть использованы с другими шеллами; и в общем применимы к Баш-скриптингу.

- Эта инструкция включает в себя стандартные Unix команды и те, для которых нужно устанавливать сторонние пакеты. Они настолько полезны, что стоят того, чтобы их установили.

Заметки:

- Для того, чтобы руководство оставалось одностраничным, вся информация вставлена прямо сюда. Вы достаточно умные для того, чтобы самостоятельно изучить вопрос более детально в другом месте. Используйте `apt-get`, `yum`, `dnf`, `pacman`, `pip` или `brew` (в зависимости от вашей системы управления пакетами) для установки новых программ.
- На Explainshell можно найти простое и подробное объяснение того, что такое команды, флаги, пайпы и т.д.

Основы

- Выучите основы Баша. Просто возьмите и напечатайте `man bash` в терминале и хотя бы просмотрите его; он довольно просто читается и он не очень большой. Другие шеллы тоже могут быть хороши, но Баш – мощная программа, и Баш всегда под рукой (использование *исключительно* zsh, fish и т.д., которые наверняка круто выглядят на Вашем ноутбуке, во многом Вас ограничивает, например, Вы не сможете использовать возможности этих шеллов на уже существующем сервере).

- Выучите как использовать хотя бы один консольный редактор текста. Идеально Vim (`vi`), ведь у него нет конкурентов, когда вам нужно быстренько что-то подправить (даже если Вы постоянно сидите на Emacs/какой-нибудь тяжелой IDE или на каком-нибудь модном хипстерском редакторе).

- Знайте как читать документацию через `man` (для любознательных – `man man`; `man` по углам документа в скобках добавляет номер, например: 1 – для обычных команд, 5 – для файлов, конвенций, 8 – для административных команд). Ищите мануалы через `apropos`, и помните, что некоторые команды – не бинарники, а встроенные команды Баша, и помощь по ним можно получить через `help` и `help -d`. С помощью `type command` можно определить, чем является команда - исполняемым файлом, алиасом или встроенной командой шелла.

- Узнайте о том, как перенаправлять ввод и вывод через `>` и `<` и пайпы `|`. Помните, что `>` переписывает выходной файл, а `>>` дописывает в конец. Узнайте побольше про stdout и stderr.

- Узнайте побольше про раскрытие file glob элементов `*` (а также `?` и `{`...`}`), кавычки, а также разницу между двойными `"` и одинарными `'` кавычками (больше о расширении переменных читайте ниже).

- Освойте управление процессами в Bash: `&`, **ctrl-z**, **ctrl-c**, `jobs`, `fg`, `bg`, `kill` и т.д.
- Знайте `ssh` и основы беспарольной аутентификации через `ssh-agent`, `ssh-add` и т.д.
- Основы работы с файлами: `ls` и `ls -l` (в частности, узнайте, что значит каждый столбец в `ls -l`), `less`, `head`, `tail` и `tail -f` (или даже лучше – `less +F`), `ln` и `ln -s` (узнайте разницу между символьными ссылками и жёсткими ссылками, и почему жёсткие ссылки лучше), `chown`, `chmod`, `du` (для быстрой сводки по использованию диска: `du -hk *`). Для менеджмента файловой системы, `df`, `mount`, `fdisk`, `mkfs`, `lsblk`. Узнайте что такое inodes (`ls -i` или `df -i`).
- Основы работы с сетью: `ip` или `ifconfig`, `dig`.
- Освойте работу с системой контроля версий, например, `git`.
- Хорошо знайте регулярные выражения и разные флаги к `grep` / `egrep`. Такие флаги, как `-i`, `-o`, `-v`, `-A`, `-B` и `-C` стоит знать.
- Обучитесь использованию систем управления пакетами `apt-get`, `yum`, `dnf` или `pacman` (в зависимости от дистрибутива), чтобы искать и устанавливать пакеты и обязательно имейте установленым `pip` для установки командных утилит, написанных на Python (некоторые из тех, что вы найдёте ниже, легче всего установить через `pip`).

Ежедневное использование

- Используйте таб в Баше для автодополнения аргументов к командам и **ctrl-r** для поиска по истории командной строки (после нажатия введите запрос, нажмите **ctrl-r** снова, чтобы найти следующее совпадение, нажмите **Enter** для выполнения текущей найденной команды или стрелку вправо, чтобы отредактировать команду).
- Используйте **ctrl-w** в Баше для того, чтобы удалить последнее слово в команде; **ctrl-u** для того, чтобы удалить команду полностью. Используйте **alt-b** и **alt-f** для того, чтобы бегать между словами команды, **ctrl-a** и **ctrl-e** для того, чтобы переместиться к началу и концу строки соответственно, **ctrl-k** для того, чтобы удалить часть команды от текущей позиции до конца строки, **ctrl-l** для того, чтобы очистить экран. Гляньте на `man readline`, чтобы узнать о всех клавиатурных сочетаниях Баша. Их много! Например, **alt-.** бежит по предыдущим аргументам команды, а **alt-*** раскрывает глоб (globbing).
- Если Вам нравятся клавиатурные сочетания vim, сделайте `set -o vi` (и `set -o emacs`, чтобы вернуться обратно).
- Для редактирования длинных команд после установки другого редактора (например `export EDITOR=vim`), нажатие **ctrl-x ctrl-e** откроет текущую команду в редакторе для многострочного редактирования. Или, как в vi, **escape-v**.

- Для просмотра последних команд используйте `history`. Повторить команду: `!n` (где `n` - порядковый номер истории). Также есть много сокращений, например, `!$` (последний аргумент) и `!!` (последняя команда) (сверьтесь со страницей man "HISTORY EXPANSION"). Впрочем, их часто проще заменить с помощью **ctrl-r** и **alt-.**.

- Перейти в домашнюю директорию можно с помощью `cd`. Для указания пути к файлам из домашней директории можно воспользоваться префиксом `~` (например, `~/.bashrc`). В `sh` скриптах для обращения к домашней директории можно использовать переменную `$HOME`.

- Для того, чтобы перейти к предыдущей рабочей директории, используйте `cd -`.

- Если вы написали команду наполовину и вдруг передумали, нажмите **alt-#** для того, чтобы добавить `#` к началу, и отправьте команду как комментарий (или используйте **ctrl-a**, **#**, **enter**). Потом вы сможете вернуться к ней через историю.

- Не забывайте использовать `xargs` (или `parallel`). Это очень мощная штука. Обратите внимание, что вы можете контролировать количество команд на каждую строку (`-L`), а также параллельность (`-P`). Если вы не уверены, что делаете что-то правильно, начните с `xargs echo`. Еще `-I{}` – полезная штука. Примеры:

```
find . -name '*.py' | xargs grep some_function
cat hosts | xargs -I{} ssh root@{} hostname
```

- `pstree -p` – полезный тип вывода дерева процессов.

- Используйте `pgrep` или `pkill` для того, чтобы находить/слать сигналы к процессам по имени (`-f` помогает).

- Знайте разные сигналы, которые можно слать процессам. Например, чтобы приостановить процесс, используйте `kill -STOP [pid]`. Для полного списка посмотрите `man 7 signal`.

- Используйте `nohup` или `disown`, чтобы процесс в фоне выполнялся бесконечно.

- Узнайте, какие процессы слушают порты через `netstat -lntp` или `ss -plat` (для TCP; добавьте `-u` для UDP).

- Используйте `lsof` для того, чтобы посмотреть открытые сокеты и файлы.

- Используйте `uptime` или `w` для того, чтобы узнать продолжительность работы системы.

- Используйте `alias`, чтобы поименовать часто используемые команды. Например, `alias ll='ls -latr'` создаст новое сокращение `ll`.

- Сохраняйте псевдонимы (aliases), настройки оболочки и часто используемые сокращения в `~/.bashrc`, и организуйте их подгрузку. Это сделает ваши настройки доступными во всех сессиях оболочки.

- Пропишите настройки переменных окружения и команды, которые должны быть выполнены при входе в систему в файл `~/.bash_profile`. Отдельная настройка будет необходима для оболочек, которые запускаются из GUI и `cron`.

- Синхронизируйте ваши конфигурационные файлы (например, `.bashrc` и `.bash_profile`) между разными компьютерами с помощью Git.

- Помните, что необходима осторожность при работе с переменными, которые содержат пробелы. Оберните свои переменные в кавычки, например `"$FOO"`. Предпочтительно использовать `-0` или `-print0` флаги, чтобы использовать нулевой символ для разделения имен файлов, например: `locate -0 pattern | xargs -0 ls -al` или `find / -print0 -type d | xargs -0 ls -al`. Для циклов, которые используют имена файлов, содержащие пробелы, установите IFS чтобы символом новой строки был только `\n`: `IFS=$'\n'`.

- В Баш-скриптах используйте `set -x` (или вариант `set -v`, который логгирует сырой ввод, включая нераскрытые переменные и комментарии) для того, чтобы отлаживать вывод (output). Используйте строгие режимы везде, где возможно. Используйте `set -e` для того, чтобы прекращать выполнение при ошибках (ненулевой код возврата). Используйте `set -u`, чтобы определять использование неинициализированных переменных. Используйте `set -o pipefail` для того, чтобы строго относиться к ошибкам (это немного глубокая тема). Для более сложных скриптов также используйте `trap` на EXIT или ERR. Полезная привычка - начинать скрипт примерно так (это поможет обнаружить ошибки и выведет предупреждение):

```
set -euo pipefail
trap "echo 'error: Script failed: see failed command above'" ERR
```

- В Баш-скриптах подоболочки (subshells) – удобный способ группировать команды. Один из самых распространенных примеров – временно передвинуться в другую рабочую директорию, вот так:

```
# do something in current dir
(cd /some/other/dir && other-command)
# continue in original dir
```

- В Баше много типов пространства переменных. Проверить, существует ли переменная – `${name:?error message}`. Например, если Баш-скрипту нужен всего один аргумент, просто напишите `input_file=${1:?usage: $0 input_file}`. Арифметическая область

видимости: `i=$(( (i + 1) % 5 ))`. Последовательности: `{1..10}`. Обрезка строк: `${var%suffix}` и `${var#prefix}`. Например, если `var=foo.pdf` тогда `echo ${var%.pdf}.txt` выведет `foo.txt`.

- Использование скобок `{...}` может уменьшить необходиость повторно вводить схожий текст и автоматизирует комбинирование элементов. Это полезно, например, здесь: `mv foo.{txt,pdf} some-dir` (переместит оба файла), `cp somefile{,.bak}` (приведется к `cp somefile somefile.bak`) или `mkdir -p test-{a,b,c}/subtest-{1,2,3}` (раскроет все возможные комбинации и создаст дерево каталогов).

- Вывод любой команды можно сохранить в файлоподобный контекст с помощью `<(some command)`. Например, сравнение локального файла `/etc/hosts с удалённым:

```
diff /etc/hosts <(ssh somehost cat /etc/hosts)
```

- Знайте про *heredoc*-синтаксис в Баше, работает он так: `cat <<EOF ...`.

- В Баше перенаправляйте стандартный вывод, а также стандартные ошибки, вот так: `some-command >logfile 2>&1` или `some-command &>logfile`. Зачастую, для того, чтобы убедиться, что команда не оставит открытым файл, привязав его к открытому терминалу, считается хорошей практикой добавлять `</dev/null`.

- Используйте `man ascii` для просмотра хорошей ASCII таблицы с шестнадцатеричными и десятичными значениями. Для информации по основным кодировкам полезны `man unicode`, `man utf-8` и `man latin1`.

- Используйте `screen` или `tmux` для того, чтобы иметь несколько экранов в одном терминале. Это особенно полезно, когда вы работаете с удаленным сервером по ssh, тогда вы можете подключаться/отключаться от сессий. `byobu` может улучшить использование `screen` или `tmux`, предоставляя больше информации и удобное управление. Более минималистичный подход для этого – использование `dtach`.

- В SSH полезно знать как сделать port tunnel с ключами `-L` и `-D` (и иногда `-R`). Например для того, чтобы зайти на сайт с удаленного сервера.

- Еще может быть полезно оптимизировать вашу SSH конфигурацию, например этот файл `~/.ssh/config` содержит настройки, которые помогают избегать потерянных подключений в некоторых сетевых окружениях. Используйте сжатие (которое полезно с scp через медленные подключения) и увеличьте количество каналов к одному серверу через этот конфиг, вот так:

```
TCPKeepAlive=yes
ServerAliveInterval=15
ServerAliveCountMax=6
```

```
    Compression=yes
    ControlMaster auto
    ControlPath /tmp/%r@%h:%p
    ControlPersist yes
```

- Некоторые другие настройки SSH могут сильно повлиять на безопасность и должны меняться осторожно, например, для конкретной подсети или конкретной машины или в доверенных сетях: `StrictHostKeyChecking=no`, `ForwardAgent=yes`.
- Рассмотрите `mosh` как альтернативу SSH, которая использует UDP и позволяет избежать разрывов соединений и добавляет удобства (требует настройки со стороны сервера).
- Чтобы получить разрешения файла в восьмеричном виде, что полезно для конфигурации систем, но нельзя получить из `ls`, можно использовать что-то типа:

```
    stat -c '%A %a %n' /etc/timezone
```

- Для интерактивного выделения результатов других команд используйте `percol` или `fzf`.
- Для работы с файлами, список которых дала другая команда (например, `git`), используйте `fpp` (PathPicker).
- Чтобы быстро поднять веб-сервер в текущей директории (и поддиректориях), который доступен для всех в вашей сети, используйте: `python -m SimpleHTTPServer 7777` (если у вас Python 2 и вы хотите открыть сервер на порту 7777) или `python -m http.server 7777` (для Python 3 и порта 7777).
- Чтобы выполнить определённую команду с привилегиями, используйте `sudo` (для рута) и `sudo -u` (для другого пользователя). Используйте `su` или `sudo bash`, чтобы запустить шелл от имени этого пользователя. Используйте `su -`, чтобы эмулировать свежий логин от рута или другого пользователя.
- Знайте про ограничение 128Кб в командной строке. Ошибка "Argument list too long" часто бывает, когда маска по имени включает большое количество файлов (в таких случаях помогают варианты с `find` или `xargs`).
- В качестве простого калькулятора (и, конечно, вообще для работы с Python) используйте интерпретатор `python`. Например,

```
>>> 2+3
5
```

Обработка файлов и информации

- Для того, чтобы найти файл в текущей директории, сделайте `find . -iname '*something*'`. Для того, чтобы искать файл по всей системе, используйте `locate something` (но не забывайте, что `updatedb` мог еще не проиндексировать недавно созданные файлы).
- Для основного поиска по содержимому файлов (более сложному, чем `grep -r`) используйте `ag`.
- Для конвертации HTML в текст: `lynx -dump -stdin`.
- Для конвертации разных типов разметки (HTML, Markdown и др.) попробуйте `pandoc`.
- Если нужно работать с XML, есть старая, но хорошая утилита – `xmlstarlet`.
- Для работы с JSON используйте `jq`.
- Для работы с YAML используйте `shyaml`.
- Для работы с Excel и CSV-файлами используйте csvkit (программа предоставляет команды `in2csv`, `csvcut`, `csvjoin`, `csvgrep` и т.д.)
- Для работы с Amazon S3 удобно работать с `s3cmd` и `s4cmd` (последний работает быстрее). Для остальных сервисов Амазона используйте стандартный `aws` или улучшенный `saws`.
- Знайте про `sort` и `uniq`, включая флаги `-u` и `-d`, смотрите примеры ниже. Также попробуйте `comm`.
- Знайте про `cut`, `paste` и `join` для работы с текстовыми файлами. Многие люди используют `cut`, забыв про `join`.
- Знайте о `wc`: для подсчёта переводов строк (`-l`), для символов – (`-m`), для слов – words (`-w`), для байтового подсчёта – (`-c`).
- Знайте про `tee` для копирования из stdin и в stdout, и в файл, например `ls -al | tee file.txt`.
- Для более сложных вычислений, включающих групповые операции с данными, преобразование матриц и статистические функции, имейте в виду `datamash`.
- Не забывайте, что локализация вашей системы влияет на многие команды, включая порядки сортировки, сравнение и производительность. Многие дистрибутивы Linux

автоматически выставляют `LANG` или любую другую переменную в подходящую для Вашего региона. Из-за этого результаты функций сортировки могут работать непредсказуемо. Рутины `i18n` могут *значительно* снизить производительность сортировок. В некоторых случаях можно полностью этого избегать (за исключением редких случаев), сортируя традиционно побайтово, для этого `export LC_ALL=C`.

- Вы можете установить специфическое окружение для команды с помощью префикса перед ее вызовом, например `TZ=Pacific/Fiji date`.
- Знайте основы `awk` и `sed` для простых манипуляций с данными. Например, чтобы получить сумму всех чисел, которые находятся в третьей колонке текстового файла, можно использовать `awk '{ x += $3 } END { print x }'`. Скорее всего, это получится раза в 3 быстрее и раза в 3 проще, чем делать это в Питоне.
- Чтобы заменить все вхождения подстроки в одном или нескольких файлах:

```
perl -pi.bak -e 's/old-string/new-string/g' my-files-*.txt
```

Для того, чтобы переименовать сразу много файлов по шаблону, используйте `rename`. Для сложных переименований может помочь `repren`. В некоторых ситуациях `rename` тоже позволяет совершать множественное переименование, но будьте осторожны, т.к. его функциональность может меняться в зависимости от дистрибутива.

```
# Full rename of filenames, directories, and contents foo -> bar:
repren --full --preserve-case --from foo --to bar .
# Recover backup files whatever.bak -> whatever:
repren --renames --from '(.*)\.bak' --to '\1' *.bak
# Same as above, using rename, if available:
rename 's/\.bak$//' *.bak
```

- Как говорит man, `rsync` на деле - быстрая, с множеством возможностей, утилита для копирования файлов. Но она хороша не только для синхронизации между машинами, но и локально. Если есть доступ, то `rsync`, в отличие от `scp`, позволяет возобновить процесс копирования, не начиная заново. Он также является самым быстрым способом удалить большое количество файлов:

```
mkdir empty && rsync -r --delete empty/ some-dir && rmdir some-dir
```

- Используйте `shuf`, чтобы перемешать строки или выбрать случайную строчку из файла.
- Знайте флаги `sort`а. Для чисел используйте `-n`, для работы с человекочитаемыми числами используйте `-h` (например `du -h`). Знайте как работают ключи (`-t` и `-k`). В

частности, не забывайте, что вам нужно писать `-k1,1` для того, чтобы отсортировать только первое поле; `-k1` - это сортировка, учитывая всю строчку. Также стабильная сортировка может быть полезной (`sort -s`). Например для того, чтобы отсортировать самое важное по второму полю, а второстепенное по первому, можно использовать `sort -k1,1 | sort -s -k2,2`.

- Если вам когда-нибудь придётся написать код символа табуляции в терминале, например, для сортировки по табуляциям с флагом -t, используйте сокращение **ctrl-v [Tab]** или напишите `$'\t'`. Последнее лучше, потому что его можно скопировать.

- Стандартные инструменты для патчинга исходников это `diff` и `patch`. Также посмотрите на `diffstat` для просмотра статистики изменений (диффа) и `sdiff` для сравнения бок-о-бок (side-by-side). `diff -r` работает рекурсивно по всей директории. Используйте `diff -r tree1 tree2 | diffstat` для полной сводки изменений. Используйте `vimdiff` для сравнения и редактирования файлов.

- Для бинарных файлов используйте `hd`, `hexdump` или `xxd` для простых hex-дампов, и `bvi` для двоичного изменения бинарных файлов.

- `strings` (в связке с `grep` или чем-то похожим) помогает найти строки в бинарных файлах.

- Чтобы посмотреть разницу в бинарниках (дельта-кодирование) используйте `xdelta3`.

- Для конвертирования кодировок используйте `iconv`. Для более сложных задач – `uconv`, он поддерживает некоторые сложные фичи Юникода. Например, эта команда переводит строки из файла в нижний регистр и убирает ударения (которые бывают, например, в испанском языке)

```
uconv -f utf-8 -t utf-8 -x '::Any-Lower; ::Any-NFD; [:Nonspacing Mark:] >; ::A
```

- Для того, чтобы разбить файл на куски, используйте `split` (разбивает на куски по размеру), или `csplit` (по шаблону или регулярному выражению).

- Для операций с датами и временем используйте `dateadd`, `datediff`, `strptime` и т.д. из `dateutils`.

- Используйте `zless`, `zmore`, `zcat` и `zgrep` для работы со сжатыми файлами.

- `chattr` устанавливает атрибуты файлов, которые также являются низкоуровневой альтернативой правам доступа. Например, от случайного удаления файла защитит атрибут неизменяемости: `sudo chattr +i /critical/directory/or/file`.

- Используйте `getfacl` и `setfacl` для сохранения и восстановления файловых прав доступа. Например:

```
getfacl -R /some/path > permissions.txt
setfacl --restore=permissions.txt
```

Отладка системы

- Для веб-отладки используйте `curl` и `curl -I`, или их альтернативу - `wget`. Также есть более современные утилиты, например `httpie`.
- Чтобы узнать текущее состояние процессора/диска, можно использовать классический `top` (или улучшенную альтернативу `htop`) и `iostat`, `iotop`. Используйте `iostat -mxz 15` для получения базовой информации о процессоре и детализированной о каждом разделе жесткого диска.
- Для получения информации о сетевых соединениях используйте `netstat` и `ss`.
- Для получения краткой информации о происходящем в системе используйте `dstat`, для более детальной информации – `glances`. Эта программа показывает сразу несколько разных статистик в одном окне терминала. Полезно, когда следите за сразу несколькими системами.
- Для того, чтобы следить за памятью, научитесь понимать `free` и `vmstat`. В частности, не забывайте, что кеширование значения ("cached" value) – это память, которую держит ядро и эти значения являются частью `free`.
- Отладка Java – совсем другая рыбка, но некоторые манипуляции над виртуальной машиной Оракла, или любой другой, позволят вам использовать `kill -3 <pid>` и трассировать сводки стека и хипа (включая детали работы сборщика мусора, которые бывают очень полезными), и их можно сдампить в stderr или логи. `jps`, `jstat`, `jstack`, `jmap` также полезны. SJK tools более продвинуты.
- Используйте `mtr` для лучшей трассировки, чтобы находить проблемы сети.
- Для того, чтобы узнать, почему диск полностью забит, используйте `ncdu`, это сохраняет время по сравнению с тем же `du -sh *`.
- Для того, чтобы узнать, какой сокет или процесс использует интернет, используйте `iftop` или `nethogs`.
- `ab`, которая поставляется вместе с apache, полезна для быстрой и поверхностной проверки производительности веб-сервера. Для более серьезного нагрузочного тестирования используйте `siege`.

- Для более серьёзной отладки сетей используйте `wireshark`, `tshark` и `ngrep`.
- Знайте про `strace` и `ltrace`. Эти команды могут быть полезны, если программа падает или висит, и вы не знаете почему. Или если вы хотите протестировать производительность программы. Не забывайте про возможность отладки (`-c`) и возможность прицепиться к процессу по pid (`-p`).
- Не забывайте про `ldd` для проверки используемых библиотек.
- Знайте как прицепиться к работающему процессу через `gdb` и получить трассировку стека.
- Используйте `/proc`. Иногда он невероятно полезен для отладки запущенных программ. Примеры: `/proc/cpuinfo`, `/proc/meminfo`, `/proc/cmdline`, `/proc/xxx/cwd`, `/proc/xxx/exe`, `/proc/xxx/fd/`, `/proc/xxx/smaps` (где `xxx` это pid).
- Когда отлаживаете что-то, что сломалось в прошлом, используйте `sar` – бывает очень полезно. Показывает историю CPU, памяти, сети и т.д.
- Для анализа более сложных систем и производительности посмотрите на `stap` (SystemTap), `perf`, и `sysdig`.
- Узнайте, какая у вас ОС, через `uname` или `uname -a` (основная Unix-информация/информация о ядре), или `lsb_release -a` (информация о дистрибутиве).
- Используйте `dmesg`, когда что-то ведет себя совсем странно (например, железо или драйвера).
- Если вы удалили файл и это вопреки ожиданиям не освободило место на диске, как показывает `du`, проверьте, использует ли файл какой-нибудь процесс: `lsof | grep deleted | grep "filename-of-my-big-file"`.

В одну строчку

Давайте соберем все вместе и напишем несколько команд:

- Это довольно круто, что можно найти множественные пересечения файлов, соединить отсортированные файлы и посмотреть разницу в нескольких файлах через `sort`/`uniq`. Это быстрый подход и работает на файлах любого размера, включая многогигабайтные файлы (сортировка не ограничена памятью, но, возможно, вам придется добавить `-T`, если `/tmp` находится на небольшом логическом диске). Еще посмотрите то, что было сказано выше о `LC_ALL`. Флаг сортировки `-u` не используется ниже, чтобы было понятнее:

```
cat a b | sort | uniq > c       # c is a union b
cat a b | sort | uniq -d > c    # c is a intersect b
cat a b b | sort | uniq -u > c  # c is set difference a - b
```

- Используйте `grep . *` для того, чтобы посмотреть содержимое всех файлов в директории. Особенно полезно, когда у вас много конфигов типа `/sys`, `/proc`, `/etc`.
- Получить сумму всех чисел, которые находятся в третьей колонке текстового файла (скорее всего, это раза в 3 быстрее и раза в 3 проще, чем делать это в Питоне):

```
awk '{ x += $3 } END { print x }' myfile
```

- Чтобы посмотреть размеры и даты в дереве файлов, есть почти как рекурсивная `ls -l`, но легче читаемая, чем `ls -lR`:

```
find . -type f -ls
```

- Скажем, у нас есть какой-то текстовый файл, например лог какого-то сервера и на каких-то строках появляется значение, строки с которым нам интересны. Например, `acct_id`. Давайте подсчитаем, сколько таких запросов в нашем логе:

```
cat access.log | egrep -o 'acct_id=[0-9]+' | cut -d= -f2 | sort | uniq -c | so
```

- Для непрерывного мониторинга измененний используйте `watch`, например, проверка измений файлов в директории: `watch -d -n 2 'ls -rtlh | tail'` или сетевых настроек во время устранения проблем с вашей wifi сетью: `watch -d -n 2 ifconfig`.
- Запустите этот скрипт, чтобы получить случайный совет из этой инструкции:

```
function taocl() {
  curl -s https://raw.githubusercontent.com/jlevy/the-art-of-command-line/mast
    pandoc -f markdown -t html |
    xmlstarlet fo --html --dropdtd |
    xmlstarlet sel -t -v "(html/body/ul/li[count(p)>0])[$RANDOM mod last()+1]"
    xmlstarlet unesc | fmt -80
}
```

Сложно, но полезно

- `expr` : для выполнения арифметических и булевых операций, а также регулярных выражений
- `m4` : простенький макро-процессор
- `yes` : вывод строки в бесконечном цикле
- `cal` : классный календарь
- `env` : для того, чтобы выполнить команду (полезно в Bash-скриптах)
- `printenv` : показать переменные окружения (полезно в скриптах или отладке)
- `look` : найти английские слова (или строки) в файле
- `cut` , `paste` и `join` : манипуляции с данными
- `fmt` : форматирование параграфов в тексте
- `pr` : отформатировать текст в страницы/колонки
- `fold` : (обернуть) ограничить длину строк в файле
- `column` : форматировать текст в колонки или таблицы
- `expand` и `unexpand` : конвертация между табами и пробелами
- `nl` : добавить номера строк
- `seq` : вывести последовательность чисел
- `bc` : калькулятор
- `factor` : возвести числа в степень
- `gpg` : зашифровать и подписать файлы
- `toe` : таблица терминалов terminfo с описанием
- `nc` : отладка сети и передачи данных
- `socat` : переключатель сокетов и перенаправление tcp-портов (похоже на `netcat`)
- `slurm` : визуализация трафика сети
- `dd` : перенос информации между блочными устройствами
- `file` : узнать тип файла

- `tree` : показать директории и поддиректории в виде дерева, как `ls` , но рекурсивно
- `stat` : информация о файле
- `time` : время выполнения команды
- `timeout` : выполнять команду указанное количество времени и остановить процесс по его истечении
- `lockfile` : создание семафорного файла, который может быть удален только с помощью `rm -f`
- `logrotate` : ротация, сжатие и отправка логов по почте
- `watch` : повторный запуск команды с выводом результата или подсветкой изменений
- `tac` : вывести файл построчно в обратном порядке
- `shuf` : случайная выборка строк из файла
- `comm` : построчно сравнить отсортированные файлы
- `pv` : мониторинг прогресса прохождения информации через пайп
- `hd` , `hexdump` , `xxd` , `biew` и `bvi` : hex-дамп и редактирование бинарных файлов
- `strings` : найти текст в бинарниках
- `tr` : манипуляция с char (символьным типом)
- `iconv` и `uconv` : конвертация кодировок
- `split` и `csplit` : разбить файлы
- `sponge` : прочитать весь входной поток перед тем, как его записать. Полезно, когда читаешь из того же файла, куда записываешь. Например, вот так: `grep -v something some-file | sponge some-file`
- `units` : конвертер. Метры в километры, версты в пяди (смотрите `/usr/share/units/definitions.units`)
- `apg` : генерация случайных паролей
- `7z` : архиватор с высокой степенью сжатия
- `ldd` : показывает зависимости программы от системных библиотек

- `nm`: получаем названия всех функций, которые определены в .o или .a (объектные файлы)
- `ab`: бенчмаркинг веб-серверов
- `strace`: отладка системных вызовов
- `mtr`: лучшая трассировка для отладки сети
- `cssh`: несколько терминалов в одном UI
- `rsync`: синхронизация файлов и папок через SSH
- `wireshark` и `tshark`: перехват пакетов и отладка сети
- `ngrep`: grep для слоя сети (network layer). Перехват пакетов по заданной маске.
- `host` и `dig`: узнать DNS
- `lsof`: информация о дескрипторах и сокетах процесса
- `dstat`: полезная статистика ОС
- `glances`: высокоуровневая статистика по многим подсистемам
- `iostat`: статистика процессора и использования жёсткого диска
- `mpstat`: статистика использования процессора
- `vmstat`: статистика использования памяти
- `htop`: улучшенная версия `top`
- `last`: история логинов в систему
- `w`: под каким пользователем вы сидите
- `id`: информация о пользователе/группе
- `sar`: история системной статистики
- `iftop` или `nethogs`: использование сети конкретным сокетом или процессом
- `ss`: статистика сокетов
- `dmesg`: ошибки загрузки и ошибки системы
- `sysctl`: просмотр и конфигурирование параметров ядра Linux

- `hdparm`: манипуляции с SATA/ATA
- `lsblk`: список блочных устройств компьютера: дерево ваших дисков и логических дисков
- `lshw`, `lscpu`, `lspci`, `lsusb`, `dmidecode`: информация о железе, включая CPU, BIOS, RAID, графику, девайсы, и т.д.
- `lsmod` и `modinfo`: информация о модулях ядра
- `fortune`, `ddate`, и `sl`: хм, не знаю, будут ли вам "полезны" веселые цитатки и поезда, пересекающие ваш терминал :)

OS X only

Некоторые вещи, подходящие *только* для OS X.

- Системы управлением пакетами – `brew` (Homebrew) и `port` (MacPorts). Они могут быть использованы для того, чтобы установить большинство программ, упомянутых в этом документе.
- Копируйте выдачу консольных программ в десктопные через `pbcopy` и вставляйте входные данные через `pbpaste`.
- Чтобы использовать в OS X кнопку Options как Alt (для использования команд **alt-b**, **alt-f** и т.д.) в настройках Терминала откройте Профили -> Клавиатура и выберите "Использовать клавишу Option в качестве метаклавиши" ("Use Option as Meta key").
- Для того, чтобы открыть файл или десктопную программу типа Finder, используйте `open`. Вот так: `open -a /Applications/Whatever.app`.
- Spotlight: Ищите файлы в консоли, через `mdfind`, и смотрите метаданные (например EXIF информацию фотографий) через `mdls`.
- Не забывайте, что OS X основана на BSD Unix и многие команды (например `ps`, `ls`, `tail`, `awk`, `sed`) имеют небольшие различия с линуксовыми. Это обусловлено влянием `UNIX System V` и `GNU Tools`. Разницу можно заметить, увидев заголовок "BSD General Commands Manual." в манах программ. В некоторых случаях, на Мак можно поставить GNU-версии программ, например `gawk` и `gsed`. Когда пишите кроссплатформенные Bash-скрипты, старайтесь избегать использовать команды, которые могут различаться (например, лучше используйте Python или `perl`), или тщательно все тестируйте.
- Чтобы получить информацию о версии OS X используйте `sw_vers`.

Windows only

- Используйте силу Unix shell в Microsoft Windows, установив Cygwin. Большая часть описанных в этом документе возможностей заработает сразу.
- Установите еще Unix программ с помощью встроенного в Cygwin менеджера пакетов.
- Используйте `mintty` в качестве терминала.
- Работайте с буфером обмена Windows с помощью `/dev/clipboard`.
- Запустите `cygstart`, чтобы открыть файл в приложении по умолчанию.
- Работайте с реестром Windows с помощью `regtool`.
- Имейте в виду, что виндовый диск `C:\` доступен в Cygwin по пути `/cygdrive/c`, и cygwin"ский `/` является папкой `C:\cygwin` в Windows. Конвертируйте файловые пути в виндовые и обратно с помощью `cygpath`. Это самый полезный скрипт, который запускает программы Windows.
- Вы можете запускать и автоматизировать большинство задач по администрированию Windows из командной строки, освоив `wmic`.

Больше информации по теме

- awesome-shell: Дополняемый список инструментов и ресурсов для командной строки.
- awesome-osx-command-line: Более детальные гайды по терминалу в OS X.
- Strict mode: Для того, чтобы писать шелл-скрипты лучше.
- shellcheck: Статический анализатор скриптов.
- Filenames and Pathnames in Shell: Сборник мелочей о правильной обработке имен файлов в скриптах.
- Data Science at the Command Line: Обзор команд и утилит, используемых для обработки данных, из одноименной книги.

Дисклеймер

За небольшим исключением, весь код написан так, чтобы другие смогли его прочитать. Чем больше сила, тем больше и ответственность. Тот факт, что вы *способны* что-то сделать в Баше, вовсе не означает, что это нужно делать! ;)

Лицензия