Notes that I've been taking while doing Unix for MacOS Users course by Kevin Skoglund

This file was created in Typora using Markdown

Contents

===========

Command examples

`echo "Hello World"` -> print text in terminal
`ruby -v` -> show version
`ruby --version` -> show version
ls -la Desktop -> list directory contents   man -ls -> format and display the on-line manual pages
echo $SHELL - > show login (default) shell
echo $0 - > show current shell
Cmd+K - clear buffer

Unix manual pages

man echo - manual page for echo
q - > exit
f ->forward
b -> backward
man --h -> manual quick overview
man -k banner - > same as apropos - searches the whatis database for strings (search in manual pages)
whatis ls - >search whatis database for complete words

Working directory

pwd -> present working directory
Listing directory contents
ls
ls -l -> vertical
ls -la -> shows hidden files
ls -lag -> shows size

Moving around filesystem

cd Desktop/ -> nav to Desktop
cd .. -> nav to parent of current dir
cd Lib+TAB -> autocomplete name of directory
cd / -> rood dir
cd ~ -> nav to user's root dir
cd -   -> nav to most recent dir

Creat files

touch somefile.txt - > change file access and modification time. also can be used to create file - creates file if it does not exist
Text editor
nano - > start editor
Ctr+X -> exit
nano newfie.txt ->

Reading files

cat shortfile.txt -> read file - one file name(all output is read for once)
cat lorem_ipsup.txt shortfile.txt -> concatenate files
more ->paginated output
less shortfile.txt -> allows to go through pages forward and backward, better memory use
q -> exit
f -> forward
b -> backward
g -> go to beginning
shift+g -> go to end
less -M shortfile.txt -> shows where currently in document
head less shortfile.txt -> display lines from beginning of a file
tail less shortfile.txt -> display lines from end of a file
tail -f shortfile.txt -> follow the tail of a file
ctr+c - > exit from head/tail

Hacks and Useful examples

tail -f /var/log/system.log -> read system log file

How to find and delete launch agents

https://discussions.apple.com/thread/7497755
turn off Adobe launcher
sudo launchctl unload -w /Library/LaunchDaemons/com.adobe.*.plist
launchctl unload -w disables each service in the override database
https://www.launchd.info/ launchd tutorial
iTerm 2 - Shell Integration

https://iterm2.com/documentation-shell-integration.html

Create directory

mkdir testdir -> create directory
mkdir -p testdir/dir2 -> create 2 directories
Moving and renaming directories
mv demofile.txt testdir/ -> move file from current dir to testdir
mv demofile.txt ../demofile.txt -> move file to parent dir
mv demofile.txt .. -> move file to parent
mv demofile.txt new_demofile.txt -> rename file
mv demofile.txt testdir/new_demofile.txt -> move and rename file
mv testdir unix_files ->rename dir
mv options
-n -> no overiting
-f -> force overwritning (default option)
-i -> interactive overwriting
-v -> verbose

Title

dfgfdg

dfgfdg

- werf

Copying files and directories

cp demofile.txt demofile2.txt -> copy from 1st to 2nd
cp options
-n -> no overiting
-f -> force overwritning (default option)
-i -> interactive overwriting
-v -> verbose

Deleting files and directories

rm demofile.txt - > delete file
rmdir delete_me -> delete dir (works for empty only)
rm -R delete_me -> delete dir and all subdirectories

Finder aliases in Unix

Aliases created in Finder cannot be used in terminal

hard-links

do not break if file is deleted
do not break if file is moved
ln linkedfile.txt hardlink -> create hard link

symbolic-links (they work in finder)

```
▶ ln -s linkedfile.txt symlink
```

lrwxr-xr-x   1 maksim staff   14 26 Nov 22:34 symlink -> linkedfile.txt

reference a file path or dir path
break if file is moved
break if file is deleted

searching-files-and-directories

find path expression
wildcards
[] - any character in the bracket

```
find / -name "index.????"
```

```
find ~/ -name *.plist
```

```
find ~ -name *.plist -and -path *QuickTime*
```

ownership-and-permissions

you can log in as user or as "root"

```
▶ whoami
```

maksim

user's home directory

```
▶ cd ~
```

```
▶ echo $HOME
```

/Users/maksim

groups

`▶ groups`
staff everyone localaccounts _appserverusr admin _appserveradm _lpadmin _appstore _lpoperator _developer _analyticsusers com.apple.access_ftp com.apple.access_screensharing com.apple.access_ssh com.apple.access_remote_ae

file-and-directory-ownership

change ownership
`▶ chown maksim:staff workflowy-export.html`
`▶ chown maksim workflowy-export.html`

admin can change ownership from other user

`▶ sudo chown maksim:staff workflowy-export.html`

file-and-directory-permissions

```
drwxr-xr-x  10 maksim  staff    320 26 Nov 21:47 .
drwxr-xr-x  13 maksim  staff    416  3 Dec 06:42 ..
-rw-r--r--@  1 maksim  staff  10244 27 Nov 16:35 .DS_Store
drwxr-xr-x  13 maksim  staff    416  7 Dec 07:10 .git
-rw-r--r--   1 maksim  staff     66 18 Nov 19:54 .gitattributes
drwxr-xr-x@  4 maksim  staff    128 29 Apr  2011 Ex_Files_UnixMacOSX
```

d - directory
"-" - file
r -read
w - write
x - execute

user: can read, write, cannot execute
staff (group): can read, cannot write and exexute
other (group): can read, cannot write and execute

`-rw-r--r--@  1 maksim  staff   5199  6 Dec 23:11 README.md`

setting-permissions-using-alpha-notation

change permission (mode)
chmod mode filename

user and group have "write" permission

`▶ chmod ug+w readME.md`
-rw-rw-r--@ 1 maksim staff   5199 6 Dec 23:11 README.md

user, group and other have read, write and execute permission
`cmod ugo=rwx filename`

disable other group writing permission
`▶ chmod o-w readme.md`

all groups: remove read, write permissions
`▶ chmod a-rw readme.md`

user, group, other have read and write permissions
`▶ chmod  ugo=rx readme.md`

add "write" permissions to folder recursively
`▶ cmod -R g+w unix_files`

setting-permissions-using-octal-notations

r - 4
w - 2
x - 1

rwxrw-r-- = 764

user, group, other have all permissions
`▶ chmod 777 demofile2.txt`
-rwxrwxrwx@ 1 maksim staff   11 5 Nov 22:11 demofile2.txt

`▶ chmod 764 demofile2.txt`
-rwxrw-r--@ 1 maksim staff   11 5 Nov 22:11 demofile2.txt

`▶ chmod 000 demofile2.txt`
----------   1 maksim staff   11 5 Nov 22:11 demofile2.txt

The root user

Superuser account can do anything on the system
Remote Unix servers usually have the root user enabled
sudo - substitute user and do
Only admins can use sudo

`▶ sudo whoami`

`Password:`

`root`

become different user
`▶ sudo -u lynda whoami`

Command basics

`▶ echo "Hello world"`

`Hello world`

Echo command is just a file located here:
/bin/echo

`▶ /bin/echo "hello"`

`hello`

`

▶ whereis echo
/bin/echo

▶ which echo
echo: shell built-in command`

Common options: -v, --version, --help
Exit: q,x,ctrl+q, ctrl+x, or ESC, or !q
Force quit: Control+c
Semicolons between commands

the-path-variable

```
▶ echo $PATH
/Library/Frameworks/Python.framework/Versions/3.8/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/X11/bin:/opt/ImageMagick/bin
```

- list, separated by colons, that unix use to locate commands to execute

To change path in bash use:
PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/X11/bin:/

setting the path only lsats for the current session (close and open terminal again)

example: this is where python executed from

```
▶ which python
/usr/bin/python
```

system-information-commands

```
▶date
Thu 19 Dec 2019 22:06:58 AEDT
```

```
▶ uptime
22:07  up 11 days, 19:32, 2 users, load averages: 2.05 2.61 2.35
```

```
▶ users
maksim
```

```
▶ who
maksim   console  Dec  8 02:35
maksim   ttys000  Dec 19 22:03
```

```
▶ uname
Darwin
```

```
▶ uname -mnrsvp
Darwin Mac-mini-Maksim.local 19.0.0 Darwin Kernel Version 19.0.0: Thu Oct 17
16:17:15 PDT 2019; root:xnu-6153.41.3~29/RELEASE_X86_64 x86_64 i386
```

```
▶ hostname
Mac-mini-Maksim.local
```

Disk information commands

disk free space:
```
▶ df
```

humanised:
```
▶ df -h
```

disk usage(amount that has bee set aside != size of file):
```
▶ du -h  ~/Dropbox
```

disk usage (folders and files):
```
▶ du -ha  ~/Dropbox
```

disk usage (1 directory deep):
```
▶ du -hd 1  ~/Dropbox
```

```
▶ du -hd 0  ~/Dropbox
1.4G    /Users/maksim/Dropbox
```

Viewing processes

process status(by default shows processes owned by user and those controlling the terminal):
```
▶ ps
  PID TTY           TIME CMD
17662 ttys000    0:00.05 /Applications/iTerm.app/Contents/MacOS/iTerm2 --server
login -fp maks
17664 ttys000    0:00.18 -zsh
```

process status(owned by others):
```
▶ ps -a
```

a - all users
u - include column showing user
x - background processes

`▶ ps aux`

Monitoring processes

Show list of top processes
`▶ top`

q - exit
-n - top 10 processes
-o - sorted by CPU usage
-s 3 - refreshed every 3 seconds
-U   - only processess of user
`▶ top -n 10 -o cpu -s 3 -U maksim`

enter "?" - display help screen
enter "s5" - updates inerval (refresh evey 5 sec)

Stopping processes

"Ctr+C" - stop process

 Show processes
`▶ ps aux`

Output:
```
maksim           54097   0.0  0.1  4298172   9320 s001  Ss    9:44pm   0:00.09
/Applications/iTerm.app/Contents/MacOS/iTerm2 --server login
```

Kill process 54097:
`▶ kill 54097`

Some processes cant be just killed. Then use "force kill"
`▶ kill -9 54097`

Text file helpers

wc - word count
sort - sort lines
uniq - filter in/out repeated lines

word count:
`▶ wc fruit.txt`
13 lines
13 words

99 characters

show top part of the text file:
`▶ head lorem_ipsum.txt`

sort output:
`▶ sort  fruit.txt`

reverse sort:
`▶ sort  -r fruit.txt`

sorted and unique:
`▶ sort -u fruit.txt`

dedupe:
`▶ uniq fruit.txt`

return repeated lines:
```
▶ uniq -d fruit.txt
strawberry
```

show unduplicated lines:
`▶ uniq -u fruit.txt`

Utility programs

cal / ncal - calendar
bc - calculator
expr - expr evaluator
units - unit conversion

`▶ cal 01 2020`

`   January 2020
Su Mo Tu We Th Fr Sa
          1 2 3 4
 5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31``

Whole year:
`▶ cal -y 2020`

Current year:
`▶ cal -y`

Days on the left:
```
▶ ncal
    January 2020
Mo      6 13 20 27
```

```
Tu      7 14 21 28
We   1  8 15 22 29
Th   2  9 16 23 30
Fr   3 10 17 24 31
Sa   4 11 18 25
Su   5 12 19 26
```

Calculator:

```
▶ bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type
```

warranty'.
1+1
2`

Set 10 decimal places:

```
scale=10
1000/9
111.1111111111
```

Quit:

```
quit
```

Simple expressions:

```
▶ expr 1 + 3
4
```

Units conversion:

```
▶ units
586 units, 56 prefixes
You have: 1 meter
You want: foot
    * 3.2808399
    / 0.3048
You have:
```

Another way:

```
▶ units '1 miles' 'kilometers'
    * 1.609344
    / 0.62137119
```

Quit:
Ctr+C

Using the command history

`` `▶ cd ~ ``

~
`▶ ls -la`

File with history:

```
-rw-------    1 maksim  staff    18727 21 Oct 10:30 .bash_history
-rw-------    1 maksim  staff       40 24 Aug 19:18 .node_repl_history
-rw-------    1 maksim  staff    22988 13 Feb 21:02 .zsh_history
```

New entries are added when session ends (quit terminal)

View history using command:

```
▶ history
    1  zsh help
    2  clear
    3  git add remote origin https://github.com/MaksimZinovev/git-basics.git
    4  git remote add origin https://github.com/MaksimZinovev/git-basics.git
    5  git remote add origin https://github.com/MaksimZinovev/git-basics.git
    6  cd/repos
```

Reference command from history

`▶ !640

~
▶ ls -la
total 11472
drwxr-xr-x+`

Go back to 2 commands ago:
`` `▶ !-10 ``

~
`▶ python3 -m pip install request`s

Start typing previous command the hit "Return":

`!py`

Output:

`▶ pytest snippets/test_create_editpdf.py`

Recal latest command:

```
▶ sudo !!

~
▶ sudo cat .zsh_history
```

Reference to the arguments of previous command. Example

```
▶ tree -man
tree: Invalid argument -`m'.
usage: tree [-acdfghilnpqrstuvxACDFJQNSUX] [-H baseHREF] [-T title ]
  [-L level [-R]] [-P pattern] [-I pattern] [-o filename] [--version]
  [--help] [--inodes] [--device] [--noreport] [--nolinks] [--dirsfirst]
  [--charset charset] [--filelimit[=]#] [--si] [--timefmt[=]<f>]
  [--sort[=]<name>] [--matchdirs] [--ignore-case] [--fromfile] [--]
  [<directory list>]

~
⍉
▶ sudo !$

# Output:
▶ cd -man

sudo: invalid option -- m
usage: sudo -h | -K | -k | -V
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user]
            [command]
usage: sudo [-AbEHknPS] [-C num] [-g group] [-h host] [-p prompt] [-T timeout]
            [-u user] [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-AknS] [-C num] [-g group] [-h host] [-p prompt] [-T timeout]
            [-u user] file ...

~
⍉
▶
```

Delete from history

```
  952  cd data
  953  tree -L 1
  954  history
  955  tree -L 1
  956  tree -man
  957  cd -man
  958  sudo -man
```

```
~
▶ history -d 952
  952  20:10  cd data
  953  20:10  tree -L 1
  954  21:32  history
  955  19:21  tree -L 1
  956  19:21  tree -man
  957  19:21  cd -man
  958  19:22  sudo -man
  959  19:24  history
```

Clear history

```
▶ history  -c
History file deleted. Reload the session to see its effects.
```

Directing input and output

stdin - standard

stdout - standard output

Direct output to a filename (only file can be on the right side)

```
▶ sort fruit.txt > sorted.txt

▶ echo "Hello world" > hello_world.txt
```

Join two files

```
cat fruit.txt hello_world.txt > joined.txt

▶ cat joined.txt
pear
raspberry
banana
peach
```

```
apple
pineapple
blueberry
papaya
strawberry
strawberry
plum
pear
apple
Hello world
```

Appending to a file (only file can be on the right side)

```
▶ echo "Mango" >> fruit.txt

Chapter_06/06_02_files/unix_files  master ✗                        96d22h ⚑ ◒
▶ cat fruit.txt
pear
raspberry
banana
peach
apple
pineapple
blueberry
papaya
strawberry
strawberry
plum
pear
apple
Mango
```

Direct input from a file (only file can be on the right side)

```
▶ sort < fruit.txt
Mango
apple
apple
banana
blueberry
papaya
peach
pear
pear
pineapple
```

```
plum
raspberry
strawberry
strawberry

#word count
▶ wc < fruit.txt
      14      14     105

 #calculate from file using basic calculator
 ▶ echo "2+2" > calc.txt

Chapter_06/06_02_files/unix_files  master ✗                          96d22h ⚑ ◒
▶ bc < calc.txt
4
```

Piping output to input

```
#word count of the string (lines / words / chars)
echo "Hello world" | wc
       1       2      12

#calc
▶ echo "2+3" |  bc
5

# sort and then remove dupl
▶ cat fruit.txt | sort | uniq
Mango
apple
banana
blueberry
papaya
peach
pear
pineapple
plum
raspberry
strawberry
```

supressing output (> /dev/null)

```
ls -la > /dev/null
```

Configuring your working environment

Upon login to a bash shell:

```
/etc/profile - is being read first
 ~/.bash_profile, ~/.bash_login, and ~/.profile, ~/,login - first file found
is being loaded, the rest is ignored
```

Upon starting a new sub shell

```
~/bashrc
```

Upon loggin out of bash shell

```
~/bash.logout
```

When an interactive shell that is not a login shell is started, Bash reads and executes commands from ~/.bashrc, if that file exists. So typically, your ~/.bash_profile contains the line

```
if [ -f ~/.bashrc ]; then . ~/.bashrc; fi
```

Let's create bash profile

```
▶ touch .bashrc

~
▶ nano .bash_profile

#paste the following code

# This only runs on user login
echo ""
echo -n "Welcome to Unix on Mac OS X, "; whoami
echo ""
echo -n "Today is "; date "+%m-%d-%Y %H:%M:%S"
echo ""
```

```
cal

# This loads in the configuration in .bashrc
# Put all configurationin ~/.bashrc!


#This code is placed in .bash_profile to load config from ~/.bashrc!!
if [ -f ~/.bashrc ]; then
  source ~/.bashrc
fi

# save and exit nano
# close terminal window
# Open new terminal window Cmd+N
# if you use iTerm2 Preferences - > general -> profiles -> commannd -> send
text at start ->  "source ~/.bash_profile"
#You can separate commands with a ; and that will allow multiple commands on
one line

#output:

Last login: Thu Apr  9 20:45:44 on ttys001

~
▶ source ~/.bash_profile

Welcome to Unix on Mac OS X, maksim

Today is 04-09-2020 20:48:48

     April 2020
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
```

Setting command aliases

Return the list of current aliases

```
▶ alias

-='cd -'
...=../..
```

```
....=../../..
.....=../../../..
......=../../../../..
1='cd -'
2='cd -2'
3='cd -3'
4='cd -4'
5='cd -5'
6='cd -6'
-='cd -'
...=../..
....=../../..
.....=../../../..
......=../../../../..
1='cd -'
2='cd -2'
3='cd -3'
4='cd -4'
5='cd -5'
6='cd -6'
7='cd -7'
8='cd -8'
9='cd -9'
_='sudo '
afind='ack -il'
egrep='egrep --color=auto --exclude-dir={.bzr,CVS,.git,.hg,.svn,.idea,.tox}'
fgrep='fgrep --color=auto --exclude-dir={.bzr,CVS,.git,.hg,.svn,.idea,.tox}'
g=git
ga='git add'
gaa='git add --all'
gap='git apply'
gapa='git add --patch'
gau='git add --update'
gav='git add --verbose'
gb='git branch'
gbD='git branch -D'
gba='git branch -a'
gbd='git branch -d'
gbda='git branch --no-color --merged | command grep -vE "^(\+|\*|\s*
(master|develop|dev)\s*$)" | command xargs -n 1 git branch -d'
gbl='git blame -b -w'
gbnm='git branch --no-merged'
gbr='git branch --remote'
gbs='git bisect'
gbsb='git bisect bad'
gbsg='git bisect good'
gbsr='git bisect reset'
gbss='git bisect start'
gc='git commit -v'
```

```
'gc!'='git commit -v --amend'
gca='git commit -v -a'
'gca!'='git commit -v -a --amend'
gcam='git commit -a -m'
'gcan!'='git commit -v -a --no-edit --amend'
'gcans!'='git commit -v -a -s --no-edit --amend'
gcb='git checkout -b'
'gc!'='git commit -v --amend'
gca='git commit -v -a'
'gca!'='git commit -v -a --amend'
gcam='git commit -a -m'
'gcan!'='git commit -v -a --no-edit --amend'
'gcans!'='git commit -v -a -s --no-edit --amend'
gcb='git checkout -b'
gcd='git checkout develop'
gcf='git config --list'
gcl='git clone --recurse-submodules'
gclean='git clean -id'
gcm='git checkout master'
gcmsg='git commit -m'
'gcn!'='git commit -v --no-edit --amend'
gco='git checkout'
gcount='git shortlog -sn'
gcp='git cherry-pick'
gcpa='git cherry-pick --abort'
gcpc='git cherry-pick --continue'
gcs='git commit -S'
gcsm='git commit -s -m'
gd='git diff'
gdca='git diff --cached'
gdct='git describe --tags $(git rev-list --tags --max-count=1)'
gdcw='git diff --cached --word-diff'
gds='git diff --staged'
gdt='git diff-tree --no-commit-id --name-only -r'
gdw='git diff --word-diff'
gf='git fetch'
gfa='git fetch --all --prune'
gfg='git ls-files | grep'
gfo='git fetch origin'
gg='git gui citool'
gga='git gui citool --amend'
ggpull='git pull origin "$(git_current_branch)"'
ggpur=ggu
ggpush='git push origin "$(git_current_branch)"'
ggsup='git branch --set-upstream-to=origin/$(git_current_branch)'
ghh='git help'
gignore='git update-index --assume-unchanged'
gignored='git ls-files -v | grep "^[[:lower:]]"'
git-svn-dcommit-push='git svn dcommit && git push github master:svntrunk'
```

```
gk='\gitk --all --branches'
gke='\gitk --all $(git log -g --pretty=%h)'
gl='git pull'
glg='git log --stat'
glgg='git log --graph'
glgga='git log --graph --decorate --all'
glgm='git log --graph --max-count=10'
glgp='git log --stat -p'
glo='git log --oneline --decorate'
globurl='noglob urlglobber '
glod='git log --graph --pretty='\''%Cred%h%Creset -%C(auto)%d%Creset %s
%Cgreen(%ad) %C(bold blue)<%an>%Creset'\'
glods='git log --graph --pretty='\''%Cred%h%Creset -%C(auto)%d%Creset %s
%Cgreen(%ad) %C(bold blue)<%an>%Creset'\'' --date=short'
glog='git log --oneline --decorate --graph'
gloga='git log --oneline --decorate --graph --all'
glol='git log --graph --pretty='\''%Cred%h%Creset -%C(auto)%d%Creset %s
%Cgreen(%cr) %C(bold blue)<%an>%Creset'\'
glola='git log --graph --pretty='\''%Cred%h%Creset -%C(auto)%d%Creset %s
%Cgreen(%cr) %C(bold blue)<%an>%Creset'\'' --all'
glols='git log --graph --pretty='\''%Cred%h%Creset -%C(auto)%d%Creset %s
%Cgreen(%cr) %C(bold blue)<%an>%Creset'\'' --stat'
glp=_git_log_prettily
glum='git pull upstream master'
gm='git merge'
gma='git merge --abort'
gmom='git merge origin/master'
gmt='git mergetool --no-prompt'
gmtvim='git mergetool --no-prompt --tool=vimdiff'
gmum='git merge upstream/master'
gp='git push'
gpd='git push --dry-run'
gpf='git push --force-with-lease'
'gpf!'='git push --force'
gpoat='git push origin --all && git push origin --tags'
gpristine='git reset --hard && git clean -dffx'
gpsup='git push --set-upstream origin $(git_current_branch)'
gpu='git push upstream'
gpv='git push -v'
gr='git remote'
gra='git remote add'
grb='git rebase'
grba='git rebase --abort'
grbc='git rebase --continue'
grbd='git rebase develop'
grbi='git rebase -i'
grbm='git rebase master'
grbs='git rebase --skip'
grep='grep --color=auto --exclude-dir={.bzr,CVS,.git,.hg,.svn,.idea,.tox}'
```

```
grev='git revert'
grh='git reset'
grhh='git reset --hard'
grm='git rm'
grmc='git rm --cached'
grmv='git remote rename'
groh='git reset origin/$(git_current_branch) --hard'
grrm='git remote remove'
grs='git restore'
grset='git remote set-url'
grss='git restore --source'
grt='cd "$(git rev-parse --show-toplevel || echo .)"'
gru='git reset --'
grup='git remote update'
grv='git remote -v'
gsb='git status -sb'
gsd='git svn dcommit'
gsh='git show'
gsi='git submodule init'
gsps='git show --pretty=short --show-signature'
gsr='git svn rebase'
gss='git status -s'
gst='git status'
gsta='git stash push'
gstaa='git stash apply'
gstall='git stash --all'
gstc='git stash clear'
gstd='git stash drop'
gstl='git stash list'
gstp='git stash pop'
gsts='git stash show --text'
gstu='git stash --include-untracked'
gsu='git submodule update'
gsw='git switch'
gswc='git switch -c'
gtl='gtl(){ git tag --sort=-v:refname -n -l "${1}*" }; noglob gtl'
gts='git tag -s'
gtv='git tag | sort -V'
gunignore='git update-index --no-assume-unchanged'
gunwip='git log -n 1 | grep -q -c "\-\-wip\-\-" && git reset HEAD~1'
gup='git pull --rebase'
gupa='git pull --rebase --autostash'
gupav='git pull --rebase --autostash -v'
gupv='git pull --rebase -v'
gwch='git whatchanged -p --abbrev-commit --pretty=medium'
gwip='git add -A; git rm $(git ls-files --deleted) 2> /dev/null; git commit --
no-verify --no-gpg-sign -m "--wip-- [skip ci]"'
history=omz_history
imgcat=/Users/maksim/.iterm2/imgcat
```

```
imgls=/Users/maksim/.iterm2/imgls
it2api=/Users/maksim/.iterm2/it2api
it2attention=/Users/maksim/.iterm2/it2attention
it2check=/Users/maksim/.iterm2/it2check
it2copy=/Users/maksim/.iterm2/it2copy
it2dl=/Users/maksim/.iterm2/it2dl
it2getvar=/Users/maksim/.iterm2/it2getvar
it2git=/Users/maksim/.iterm2/it2git
it2setcolor=/Users/maksim/.iterm2/it2setcolor
it2setkeylabel=/Users/maksim/.iterm2/it2setkeylabel
it2ul=/Users/maksim/.iterm2/it2ul
it2universion=/Users/maksim/.iterm2/it2universion
l='ls -lah'
la='ls -lAh'
ll='ls -lh'
ls='ls -G'
lsa='ls -lah'
md='mkdir -p'
rd=rmdir
run-help=man
which-command=whence
```

create alias

```
▶ alias ll='ls -la'
```

create file with zsh custom aliases

```
▶ touch ~/.oh-my-zsh/custom/aliases.zsh
#add the following to the file
alias reload='source ~/.zshrc'
#This means I can clone a repo, then just type y to pull in all the
dependencie$
alias y='yarn'
#prints your current public IP address to the termina
alias myip='curl http://ipecho.net/plain; echo'
#output information about your Linux distribution
alias distro='cat /etc/*-release'
alias al='nano ~/.oh-my-zsh/custom/aliases.zsh'
```

create file with vbash custom aliases:

```
nano ~/.bashrc
#edit file
alias ll='ls -la'
#make parent directory
alias mkdir='mkdir -p'
alias pdw='pwd'
#load from config file "".bashrc" now
alias sbr='source ~/.bashrc'
alias cdr='cd ~/repos'
```

Zsh most useful commands

- Entering `cd` from anywhere on the file system will bring you straight back to your home directory.
- Entering `!!` will bring up the last command. This is handy if a command fails because it needs admin rights. In this case you can type `sudo !!`.
- You can use `&&` to chain multiple commands. For example, `mkdir project && cd project && npm init -y`.
- Conditional execution is possible using `||`. For example, `git commit -m "whatever..." || echo "Commit failed"`.
- Using a `-p` switch with the `mkdir` command will allow you to create parent directories as needed. Using brace expansion reduces repetition. For example, `mkdir -p articles/jim/sitepoint/article{1,2,3}`.
- Set environment variables on a per-command basis like so: `NODE_DEBUG=myapp node index.js`. Or, on a per-session basis like so: `export NODE_DEBUG=myapp`. You can check it was set by typing `echo $`.
- Pipe the output of one command into a second command. For example, `cat /var/log/kern.log | less` to make a long log readable, or `history | grep ssh` to search for any history entries containing "ssh".
- You can open files in your editor from the terminal. For example, `nano ~/.zshrc` (nano), `subl ~/.zshrc` (Sublime Text), `code ~/.zshrc` (VS Code). If the file doesn't exist, it will be created when you press *Save* in the editor.
- Navigation is an important skill to master. Don't just rely on your arrow keys. For example, Ctrl + a will take you to the beginning of a line.
- Whereas Ctrl + e will take you to the end.
- You can use Ctrl + w to delete one word (backwards).
- Ctrl + u will remove everything from the cursor to the beginning of the line.
- Ctrl + k will clear everything from the cursor to the end of the line. These last three can be undone with Ctrl + y.
- You can copy text with Ctrl + Shift + c. This is much more elegant than right clicking and

selecting *Copy*.
- Conversely, you can paste copied text with Ctrl + shift + v.

- The `take` command will create a new directory *and* change into it. `take my-project` replaces `mkdir my-project && cd my-project`.
- `zsh_stats` will give you a list of the top 20 commands and how many times they've been run.
- Oh My Zsh simplifies navigating your file system. For example, `..` is an alias for `cd ..`.
- In the same way, `...` moves you up two directories, `....` moves you up three, and `.....` moves you up four.
- You can omit the `cd` when navigating. Typing `/`, for example, will take you straight to your filesystem root.
- Partial matching is also supported. For example, typing `/h/j/De` and pressing TAB, then Return, takes me to `/home/jim/Desktop/`.
- `rd` is an alias for `rmdir` and `md` is an alias for `mkdir -p`.
- You can type `d` to list the last used directories from a terminal session.
- You can then navigate to any of these using `cd -n`, where `n` is the directory number.
- Tab completion is another great feature. For example, typing `ls -` and pressing TAB will list all of the command's options, along with a helpful description of what they do. This also works for `cap`, `rake`, `ssh`, and `kill`.
- Typing `alias` lists all of your current aliases.
- With globbing (a Zsh feature), you can list files with a particular extension. For example, `ls *.html` will list all HTML files in the current directory. To include subdirectories, change to: `ls **/*.html`.
- Glob qualifiers allow you to select types of files by using flags. For example, `ls -l **/*(.x)` will find all executable files in the current directory and all sub-directories.
- You can search for files by date modified. For example, `ls *(m-7)` will list all files modified within the last week.
- You can search for files by size. For example, `ls *(Lm+1)` will find all files with a size larger than 1MB.

Environment variables

Sreturn default login shell for the current user

```
echo $SHELL
```

"$" - indicate for the unix that we want to return the value that stored in shell variable

We can define our own shell variable

```
MYNAME='Maksim Zinovev'
echo $MYNAME
#output
Maksim Zin
```

When we logout of the current session variabbles disappear. To store them we need to save them in ~/.bashrc

```
nano ~/.bashrc
MYNAME='Maksim Zinovev'
#save file and exit shell
#open new shell window
echo $MYNAME
#output
Maksim Zinovev
```

However those variable will not be available for child processes - they will be available in bash itself. To make them available to other commands, programs and sripts we need to use "EXPORT"

```
#.bashrc
MYNAME='Maksim Zinovev'
export MYNAME
```

Export can also be used to set configuration options for our unix environment

```
#.bashrc
# Medium verbose prompt when we use 'less' command
export LESS='-m'

▶ echo $LESS
-M
```

Installing oh my zsh plugin

```
# ~/.zshrc
#download pluging from github to the folder "~/z.sh/z.sh"
plugins=(z zsh-autosuggestions)
source  ~/z.sh/z.sh

#reload
#move around
```

Setting the PATH variables

$PATH is the list of file paths which unix uses to locate commands (separated by ":"). Unix uses the order in which paths defined to look for commands

- first "/Library/Frameworks/Python.framework/Versions/3.8/bin"
- second "/usr/local/bin"
- ...

```
▶ echo $PATH
/Library/Frameworks/Python.framework/Versions/3.8/bin:/usr/local/bin:/usr/bin:
/bin:/usr/sbin:/sbin:/opt/X11/bin:/opt/ImageMagick/bin

```

We can use existing $PATH to modify it in this way (**make sure you use double quotes**)

```
#.bashrc
export PATH=≈
""/usr/local/bin:$PATH"
```

Configuring history with variables

```
#.bashrc
#number of recent commands stored in history
export HISTSIZE=10000                          # 500 is default
#set file size limit
export HISTFILESIZE=1000000
#adding timestamp
export HISTTIMEFORMAT='%b %d %I:%M %p '     # using strftime format
```

```
#ignore dups and space(any line begins with space)
export HISTCONTROL=ignoreboth              # ignoredups:ignorespace
#ignore certain commands
export HISTIGNORE="history:pwd:exit:df:ls:ls -la:ll"
```

Customizing the command prompt

```
▶ PS1="-->"
-->

▶ PS1="\u"
```

- \u - username
- \s - current shell
- \w - current working directory
- \W - basename of current working directory
- \d - date
- \D(format) - date in strftime format

Logout file

Executed every time you logout

```
nano .bash_logout
#.bash_logout
echo "See you later"

#save
exit
```

Unix power tools

Searching for matching expressions

- grep - searching with reular expressions
- **G**lobal **R**egular **E**xpresson **P**rint

Returns lines

```
▶ grep appl fruit.txt

#output
apple
pineapple
apple
```

case insencitive option

```
▶ grep -i appl fruit.txt
apple
pineapple
apple

```

search matches of whole word

```
▶ grep -w apple fruit.txt
apple
apple
```

lines that do not match

```
▶ grep -v apple fruit.txt
pear
raspberry
banana
peach
blueberry
papaya
strawberry
strawberry
plum
pear
```

count matches

```
▶ grep -c apple fruit.txt
3
```

search multiple files and other directories. Search in all files "Downloads" folder

```
▶ grep -R apple ~/Downloads
Binary file /Users/maksim/Downloads/Acrobat_DC_Installer.dmg matches
Binary file /Users/maksim/Downloads/NTS Radio - Secretsundaze & Eliphino   5th
September 2019.m4a matches
Binary file /Users/maksim/Downloads/Typora.dmg matches
Binary file /Users/maksim/Downloads/mac-video-converter-ultimate.dmg matches
/Users/maksim/Downloads/automate_online-materials/picnicTable.py:picnicItems =
{'sandwiches': 4, 'apples': 12, 'cups': 4, 'cookies': 8000}
```

list just filenames

```
▶ grep -Rl apple ~/repos/02-unix-macos
```

using grep with pipe

```
▶ ps aux | grep Terminal
maksim           70210   0.0  0.3  4919532  26816   ??  S     3:54pm   0:01.72
/System/Applications/Utilities/Terminal.app/Contents/MacOS/Terminal
maksim           84564   0.0  0.0  4295928    712 s000  S+    9:12pm   0:00.01
grep --color=auto --exclude-dir=.bzr --exclude-dir=CVS --exclude-dir=.git --
exclude-dir=.hg --exclude-dir=.svn --exclude-dir=.idea --exclude-dir=.tox
Terminal
```

list last commands with nano

```
▶ history | grep nano | less
```

highlight search term

```
▶ grep --color lorem  lorem_ipsum.txt
```

save settings in .bashrc to have search term always highlighted automatically

```
#.bashrc
export GREP_OPTIONS="--color=auto"
# now just run
▶ grep lorem  lorem_ipsum.txt
```

usign grep with regular expressions

```
▶ grep 'apple' fruit.txt
apple
pineapple
apple
```

periods(any characters) in regex

```
▶ grep 'a..le' fruit.txt
apple
pineapple
apple

▶ grep '.a.a.a' fruit.txt
banana
papaya
```

brakets mean match "c" OR "p"

```
▶ grep 'ea[cp]' fruit.txt
peach
pineapple
```

Other regex expressions

Regex	Meaning	
.	Wild card, any one character	gre.t
[ ]	character set	gr[ea]y
[^]	negative character set	[^aei]
-	range indicator	[A-Z]
*	preceding element can occur zero or more times	file_*name
+	preceding element can occur one or more times	gro+ve
?	preceding element can occur zero or one time	colou?r
|	alternation, OR operator	(jpg|gif|png)
^	start of line	^Hello
$	end of line	World$
\	escape the next character	image\.jpg
\d	any digit	20\d\d-06-09
\D	anything not a digit	^\D+
\w	any word char	\w+_export\.sql
\W	anything not a word char	\w+\W\w+
\s	whitespace	\w+\s\w+
\S	anything not whitespace	\S+s\S+

Regex character classes

class	
[:alpha:]	alphabetic characters
[:digit:]	numeric characters

Examples

```
▶ grep '^p' fruit.txt
pear
peach
pineapple
papaya
plum
pear
```

```
▶ grep 'berry$' fruit.txt
raspberry
blueberry
strawberry
strawberry
```

```
▶ echo 'AaBbCcDdEe' | grep --color [:upper:]
zsh: no matches found: [:upper:]
```

```
▶ echo 'AaBbCcDdEe' | grep --color '[[:upper:]]'
```

output

Aa**B**b**C**c**D**d**E**e

```
▶ grep 'ap+le' fruit.txt
none
```

```
▶ grep -E 'ap+le' fruit.txt
apple
pineapple
apple
```

```
▶ grep -E 'apple|pear' fruit.txt
pear
apple
pineapple
pear
apple
```

Translating (replacing) characters

```
▶ echo 'a,b,c,d' | tr ',' '-'
a-b-c-d
```

Mapping in replacment

```
▶ echo '12344543454' | tr '123456' 'EBGDAE'
EBGDDADGDAD
```

```
▶ echo 'This is ROT-13 encrypted.' | tr 'A-Za-z' 'N-ZA-Mn-za-m'
Guvf vf EBG-13 rapelcgrq.
```

More examples

```
▶ echo 'Guvf vf EBG-13 rapelcgrq.' | tr 'A-Za-z' 'N-ZA-Mn-za-m'
This is ROT-13 encrypted.
```

```
▶ tr 'A-Z' 'a-z' < people.txt
kevin
lynda
bob
susan
larry
anne
claire
john
```