Apple devices

- Mac OsX
  - Osx CLI
  - Créer une clé USB bootable Osx
- Ubuntu on a MacBookPro 8,2
- Qemu for Mac M1
- DVDrip HandBrake Mac M1

Mac OsX

Mac OsX

Osx CLI

Awesome macOS Command Lin

> A curated list of shell commands and tools specific to OS X.
>
> *"You don't have to know everything. You simply need to know where to find it when necessary." (John Brunner)*

Awesome Build Status

Appearance

Transparency

Transparency in Menu and Windows

```
# Reduce Transparency
defaults write com.apple.universalaccess reduceTransparency -bool true

# Restore Default Transparency
defaults write com.apple.universalaccess reduceTransparency -bool false
```

Wallpaper

Set Wallpaper

```
# Up to Mountain Lion
osascript -e '
tell application "Finder" to set desktop picture to POSIX file "/path/to/picture.jpg"'

# Since Mavericks
sqlite3 ~/Library/Application\ Support/Dock/desktoppicture.db "
update data set value = '/path/to/picture.jpg'" && killall Dock
```

Applications

App Store

List All Apps Downloaded from App Store

```
# Via find
find /Applications -path '*Contents/_MASReceipt/receipt' -maxdepth 4 -print |\sed '
s#.app/Contents/_MASReceipt/receipt#.app#g; s#/Applications/##'

# Via Spotlight
mdfind kMDItemAppStoreHasReceipt=1
```

Show Debug Menu

Works up to Yosemite.

```
# Enable
defaults write com.apple.appstore ShowDebugMenu -bool true

# Disable (Default)
defaults write com.apple.appstore ShowDebugMenu -bool false
```

Apple Remote Desktop

Kickstart Manual Pages

```
sudo /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kickstart -he
```

Activate And Deactivate the ARD Agent and Helper

```
# Activate And Restart the ARD Agent and Helper
sudo /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kickstart -ac

# Deactivate and Stop the Remote Management Service
sudo /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kickstart -de
```

Enable and Disable Remote Desktop Sharing

```
# Allow Access for All Users and Give All Users Full Access
sudo /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kickstart -cc

# Disable ARD Agent and Remove Access Privileges for All Users
sudo /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kickstart -de
```

Remove Apple Remote Desktop Settings

```
sudo rm -rf /var/db/RemoteManagement ; \
sudo defaults delete /Library/Preferences/com.apple.RemoteDesktop.plist ; \
defaults delete ~/Library/Preferences/com.apple.RemoteDesktop.plist ; \
sudo rm -r /Library/Application\ Support/Apple/Remote\ Desktop/ ; \
rm -r ~/Library/Application\ Support/Remote\ Desktop/ ; \
rm -r ~/Library/Containers/com.apple.RemoteDesktop
```

Contacts

Debug Mode

```
# Enable
defaults write com.apple.addressbook ABShowDebugMenu -bool true

# Disable (Default)
defaults write com.apple.addressbook ABShowDebugMenu -bool false
```

Google

Uninstall Google Update

```
~
/Library/Google/GoogleSoftwareUpdate/GoogleSoftwareUpdate.bundle/Contents/Resources/ksinstall --
```

iTunes

Keyboard Media Keys

This works up to Yosemite. System Integrity Protection was introduced in El Capitan which prevents system Launch Agents from being unloaded.

```
# Stop Responding to Key Presses
launchctl unload -w /System/Library/LaunchAgents/com.apple.rcd.plist

# Respond to Key Presses (Default)
launchctl load -w /System/Library/LaunchAgents/com.apple.rcd.plist
```

From El Capitan onwards, you can either disable SIP or resort to a kind of hack, which will make iTunes inaccessible to any user, effectively preventing it from starting itself or its helpers. Be aware that for all intents and purposes this will trash your iTunes installation and may conflict with OS updates down the road.

```
sudo chmod 0000 /Applications/iTunes.app
```

Mail

Show Attachments as Icons

```
defaults write com.apple.mail DisableInlineAttachmentViewing -bool yes
```

Vacuum Mail Index

The AppleScript code below will quit Mail, vacuum the SQLite index, then re-open Mail. On a large email database that hasn't been optimized for a while, this can provide significant improvements in responsiveness and speed.

```
(*
Speed up Mail.app by vacuuming the Envelope Index
Code from: http://web.archive.org/web/20071008123746/http://www.hawkwings.net/2007/03/03/scripts

Originally by "pmbuko" with modifications by Romulo
Updated by Brett Terpstra 2012
Updated by Mathias Törnblom 2015 to support V3 in El Capitan and still keep backwards compatibil

Updated by Andrei Miclaus 2017 to support V4 in Sierra
*)

tell application "Mail" to quit
set os_version to do shell script "sw_vers -productVersion"
set mail_version to "V2"
considering numeric strings
    if "10.10" <= os_version then set mail_version to "V3"
    if "10.12" <= os_version then set mail_version to "V4"
    if "10.13" <= os_version then set mail_version to "V5"
    if "10.14" <= os_version then set mail_version to "V6"
end considering

set sizeBefore to do shell script "ls -lnah ~/Library/Mail/" & mail_version & "
/MailData | grep -E 'Envelope Index$' | awk {'print $5'}"
do shell script "/usr/bin/sqlite3 ~/Library/Mail/" & mail_version & "/MailData/Envelope\\
 Index vacuum"

set sizeAfter to do shell script "ls -lnah ~/Library/Mail/" & mail_version & "
/MailData | grep -E 'Envelope Index$' | awk {'print $5'}"

display dialog ("Mail index before: " & sizeBefore & return & "Mail index after: " & sizeAfter
& return & return & "Enjoy the new speed!")

tell application "Mail" to activate
```

Safari

Change Default Fonts

```
defaults write com.apple.Safari com.apple.Safari.ContentPageGroupIdentifier.WebKit2StandardFontF
defaults write com.apple.Safari com.apple.Safari.ContentPageGroupIdentifier.WebKit2DefaultFontSi
defaults write com.apple.Safari com.apple.Safari.ContentPageGroupIdentifier.WebKit2FixedFontFami
defaults write com.apple.Safari com.apple.Safari.ContentPageGroupIdentifier.WebKit2DefaultFixedF
```

Enable Develop Menu and Web Inspector

```
defaults write com.apple.Safari IncludeInternalDebugMenu -bool true && \
defaults write com.apple.Safari IncludeDevelopMenu -bool true && \
defaults write com.apple.Safari WebKitDeveloperExtrasEnabledPreferenceKey -bool true && \
defaults write com.apple.Safari com.apple.Safari.ContentPageGroupIdentifier.WebKit2DeveloperExtr
true && \
defaults write -g WebKitDeveloperExtras -bool true
```

Get Current Page Data

Other options: `get source`, `get text`.

```
osascript -e 'tell application "Safari" to get URL of current tab of front window'
```

Use Backspace/Delete to Go Back a Page

```
# Enable
defaults write com.apple.Safari com.apple.Safari.ContentPageGroupIdentifier.WebKit2BackspaceKeyN

# Disable
defaults write com.apple.Safari com.apple.Safari.ContentPageGroupIdentifier.WebKit2BackspaceKeyN
```

Sketch

Export Compact SVGs

```
defaults write com.bohemiancoding.sketch3 exportCompactSVG -bool yes
```

Skim

Turn Off Auto Reload Dialog

Removes the dialog and defaults to auto reload.

```
defaults write -app Skim SKAutoReloadFileUpdate -boolean true
```

Terminal

Focus Follows Mouse

```
# Enable
defaults write com.apple.Terminal FocusFollowsMouse -string YES

# Disable
defaults write com.apple.Terminal FocusFollowsMouse -string NO
```

TextEdit

Use Plain Text Mode as Default

```
defaults write com.apple.TextEdit RichText -int 0
```

Visual Studio Code

Fix VSCodeVim Key Repeat

```
defaults write com.microsoft.VSCode ApplePressAndHoldEnabled -bool false
```

Backup

Time Machine

Change Backup Interval

This changes the interval to 30 minutes. The integer value is the time in seconds.

```
sudo defaults write /System/Library/LaunchDaemons/com.apple.backupd-auto StartInterval -int 1800
```

Local Backups

Whether Time Machine performs local backups while the Time Machine backup volume is not available.

```
# Status
defaults read /Library/Preferences/com.apple.TimeMachine MobileBackups

# Enable (Default)
sudo tmutil enablelocal

# Disable
sudo tmutil disablelocal
```

Since High Sierra, you cannot disable local snapshots. Time Machine now always creates a local APFS snapshot and uses that snapshot as the data source to create a regular backup, rather than using the live disk as the source, as is the case with HFS formatted disks.

Prevent Time Machine from Prompting to Use New Hard Drives as Backup Volume

```
sudo defaults write /Library/Preferences/com.apple.TimeMachine DoNotOfferNewDisksForBackup -bool
true
```

Show Time Machine Logs

This little script will output the last 12 hours of Time Machine activity followed by live activity.

```
#!/bin/sh

filter='processImagePath contains "backupd" and subsystem beginswith "com.apple.TimeMachine"'

# show the last 12 hours
start="$(date -j -v-12H +'%Y-%m-%d %H:%M:%S')"

echo ""
echo "[History (from $start)]"
echo ""

log show --style syslog --info --start "$start" --predicate "$filter"

echo ""
echo "[Following]"
echo ""

log stream --style syslog --info --predicate "$filter"
```

Toggle Backup While on Battery

```
# Status
sudo defaults read /Library/Preferences/com.apple.TimeMachine RequiresACPower

# Enable (Default)
sudo defaults write /Library/Preferences/com.apple.TimeMachine RequiresACPower -bool true

# Disable
sudo defaults write /Library/Preferences/com.apple.TimeMachine RequiresACPower -bool false
```

Verify Backup

Beginning in OS X 10.11, Time Machine records checksums of files copied into snapshots. Checksums are not retroactively computed for files that were copied by earlier releases of OS X.

```
sudo tmutil verifychecksums /path/to/backup
```

Developer

Vim

Compile Sane Vim

Compiling MacVim via Homebrew with all bells and whistles, including overriding system Vim.

```
brew install macvim --HEAD
```

Neovim

Install the modern Vim drop-in alternative via Homebrew.

```
brew install neovim
```

Xcode

Install Command Line Tools without Xcode

```
xcode-select --install
```

Remove All Unavailable Simulators

```
xcrun simctl delete unavailable
```

Dock

Add a Stack with Recent Applications

```
defaults write com.apple.dock persistent-others -array-add '
{ "tile-data" = { "list-type" = 1; }; "tile-type" = "recents-tile"; }' && \
killall Dock
```

Add a Nameless Stack Folder and Small Spacer

```
defaults write com.apple.dock persistent-others -array-add '
{ "tile-data" = {}; "tile-type"="small-spacer-tile"; }' && \
killall Dock
```

Add a Space

```
defaults write com.apple.dock persistent-apps -array-add '{"tile-type"="spacer-tile";}' && \
killall Dock
```

Add a Small Space

```
defaults write com.apple.dock persistent-apps -array-add '{"tile-type"="small-spacer-tile";}'
&& \
killall Dock
```

Auto Rearrange Spaces Based on Most Recent Use

```
# Enable (Default)
defaults write com.apple.dock mru-spaces -bool true && \
killall Dock

# Disable
defaults write com.apple.dock mru-spaces -bool false && \
killall Dock
```

Autohide

```
# Enable
defaults write com.apple.dock autohide -bool true && \
killall Dock

# Disable (Default)
defaults write com.apple.dock autohide -bool false && \
killall Dock
```

Icon Bounce

Global setting whether Dock icons should bounce when the respective application demands your attention.

```
# Enable (Default)
defaults write com.apple.dock no-bouncing -bool true && \
killall Dock

# Disable
defaults write com.apple.dock no-bouncing -bool false && \
killall Dock
```

Lock the Dock Size

```
# Enable
defaults write com.apple.Dock size-immutable -bool yes && \
killall Dock
```

```
# Disable (Default)
defaults write com.apple.Dock size-immutable -bool no && \
killall Dock
```

Reset Dock

```
defaults delete com.apple.dock && \
killall Dock
```

Resize

Fully resize your Dock's body. To resize change the `0` value as an integer.

```
defaults write com.apple.dock tilesize -int 0 && \
killall Dock
```

Scroll Gestures

Use your touchpad or mouse scroll wheel to interact with Dock items. Allows you to use an upward scrolling gesture to open stacks. Using the same gesture on applications that are running invokes Exposé/Mission Control.

```
# Enable
defaults write com.apple.dock scroll-to-open -bool true && \
killall Dock

# Disable (Default)
defaults write com.apple.dock scroll-to-open -bool false && \
killall Dock
```

Set Auto Show/Hide Delay

The float number defines the show/hide delay in ms.

```
defaults write com.apple.dock autohide-time-modifier -float 0.4 && \
defaults write com.apple.dock autohide-delay -float 0 && \
killall Dock
```

Show Hidden App Icons

```
# Enable
defaults write com.apple.dock showhidden -bool true && \
killall Dock

# Disable (Default)
defaults write com.apple.dock showhidden -bool false && \
killall Dock
```

Show Only Active Applications

```
# Enable
defaults write com.apple.dock static-only -bool true && \
killall Dock

# Disable (Default)
defaults write com.apple.dock static-only -bool false && \
killall Dock
```

Single App Mode

When clicking an application icon in the Dock, the respective windows will come to the front, but all other application windows will be hidden.

```
# Enable
defaults write com.apple.dock single-app -bool true && \
killall Dock

# Disable (Default)
defaults write com.apple.dock single-app -bool false && \
killall Dock
```

Documents

Convert File to HTML

Supported formats are plain text, rich text (rtf) and Microsoft Word (doc/docx).

```
textutil -convert html file.ext
```

Files, Disks and Volumes

Create an Empty File

Creates an empty 10 gigabyte test file.

```
mkfile 10g /path/to/file
```

Disable Sudden Motion Sensor

Leaving this turned on is useless when you're using SSDs.

```
sudo pmset -a sms 0
```

Eject All Mountable Volumes

The only reliable way to do this is by sending an AppleScript command to Finder.

```
osascript -e 'tell application "Finder" to eject (every disk whose ejectable is true)'
```

Repair File Permissions

You don't have to use the Disk Utility GUI for this.

```
sudo diskutil repairPermissions /
```

> Beginning with OS X El Capitan, system file permissions are automatically protected. It's no longer necessary to verify or repair permissions with Disk Utility. (Source)

Set Boot Volume

```
# Up to Yosemite
bless --mount "/path/to/mounted/volume" --setBoot

# From El Capitan
sudo systemsetup -setstartupdisk /System/Library/CoreServices
```

Show All Attached Disks and Partitions

```
diskutil list
```

View File System Usage

A continuous stream of file system access info.

```
sudo fs_usage
```

APFS

Available since High Sierra. There is no central utility and usage is inconsistent as most functionality is rolled into `tmutil`.

Convert Volume from HFS+ to APFS

```
/System/Library/Filesystems/apfs.fs/Contents/Resources/hfs_convert /path/to/file/system
```

Create New APFS Filesystem

```
/System/Library/Filesystems/apfs.fs/Contents/Resources/newfs_apfs /path/to/device
```

Create Snapshot

```
tmutil localsnapshot
```

Delete Snapshot

```
tmutil deletelocalsnapshots com.apple.TimeMachine.2018-01-26-044042
```

List Snapshots

```
tmutil listlocalsnapshots /
```

Mount Snapshot

Snapshots are read-only.

```
mkdir ~/mnt
/System/Library/Filesystems/apfs.fs/Contents/Resources/mount_apfs -s com.apple.TimeMachine.2018-
~/mnt
```

Disk Images

Create Disk Image From Folder Contents

```
hdiutil create -volname "Volume Name" -srcfolder /path/to/folder -ov diskimage.dmg
```

If you'd like to encrypt the disk image:

```
hdiutil create -encryption -stdinpass -volname "Volume Name"
 -srcfolder /path/to/folder -ov encrypted.dmg
```

By default, you'll be prompted for a password. You can automate that by piping in a password:

```
echo -n YourPassword | hdiutil create -encryption -stdinpass -volname "Volume Name"
 -srcfolder /path/to/folder -ov encrypted.dmg
```

Burn Disk Images to DVD

This command applies to .iso, .img and .dmg images.

```
hdiutil burn /path/to/image_file
```

Disable Disk Image Verification

```
defaults write com.apple.frameworks.diskimages skip-verify -bool true && \
defaults write com.apple.frameworks.diskimages skip-verify-locked -bool true && \
```

```
defaults write com.apple.frameworks.diskimages skip-verify-remote -bool true
```

Make Volume OS X Bootable

```
bless --folder "/path/to/mounted/volume/System/Library/CoreServices" --bootinfo --bootefi
```

Mount Disk Image

```
hdiutil attach /path/to/diskimage.dmg
```

Unmount Disk Image

```
hdiutil detach /dev/disk2s1
```

Write Disk Image to Volume

Like the Disk Utility "Restore" function.

```
sudo asr -restore -noverify -source /path/to/diskimage.dmg -target /Volumes/VolumeToRestoreTo
```

Finder

Desktop

Show External Media

External HDs, thumb drives, etc.

```
# Enable
defaults write com.apple.finder ShowExternalHardDrivesOnDesktop -bool true && \
killall Finder

# Disable (Default)
defaults write com.apple.finder ShowExternalHardDrivesOnDesktop -bool false && \
killall Finder
```

Show Internal Media

Built-in HDs or SSDs.

```
# Enable
defaults write com.apple.finder ShowHardDrivesOnDesktop -bool true && \
killall Finder

# Disable (Default)
```

```
defaults write com.apple.finder ShowHardDrivesOnDesktop -bool false && \
killall Finder
```

Show Removable Media

CDs, DVDs, iPods, etc.

```
# Enable
defaults write com.apple.finder ShowRemovableMediaOnDesktop -bool true && \
killall Finder

# Disable (Default)
defaults write com.apple.finder ShowRemovableMediaOnDesktop -bool false && \
killall Finder
```

Show Network Volumes

AFP, SMB, NFS, WebDAV, etc.

```
# Enable
defaults write com.apple.finder ShowMountedServersOnDesktop -bool true && \
killall Finder

# Disable (Default)
defaults write com.apple.finder ShowMountedServersOnDesktop -bool false && \
killall Finder
```

Files and Folders

Clear All ACLs

```
sudo chmod -RN /path/to/folder
```

Hide Folder in Finder

```
chflags hidden /path/to/folder/
```

Show All File Extensions

```
defaults write -g AppleShowAllExtensions -bool true
```

Show Hidden Files

```
# Show All
defaults write com.apple.finder AppleShowAllFiles true

# Restore Default File Visibility
```

```
defaults write com.apple.finder AppleShowAllFiles false
```

Remove Protected Flag

```
sudo chflags -R nouchg /path/to/file/or/folder
```

Show Full Path in Finder Title

```
defaults write com.apple.finder _FXShowPosixPathInTitle -bool true
```

Unhide User Library Folder

```
chflags nohidden ~/Library
```

Increase Number of Recent Places

```
defaults write -g NSNavRecentPlacesLimit -int 10 && \
killall Finder
```

Layout

Show "Quit Finder" Menu Item

Makes possible to see Finder menu item "Quit Finder" with default shortcut `Cmd` + `Q`

```
# Enable
defaults write com.apple.finder QuitMenuItem -bool true && \
killall Finder

# Disable (Default)
defaults write com.apple.finder QuitMenuItem -bool false && \
killall Finder
```

Smooth Scrolling

Useful if you’re on an older Mac that messes up the animation.

```
# Disable
defaults write -g NSScrollAnimationEnabled -bool false

# Enable (Default)
defaults write -g NSScrollAnimationEnabled -bool true
```

Rubberband Scrolling

```
# Disable
defaults write -g NSScrollViewRubberbanding -bool false
```

```
# Enable (Default)
defaults write -g NSScrollViewRubberbanding -bool true
```

Expand Save Panel by Default

```
defaults write -g NSNavPanelExpandedStateForSaveMode -bool true && \
defaults write -g NSNavPanelExpandedStateForSaveMode2 -bool true
```

Desktop Icon Visibility

```
# Hide Icons
defaults write com.apple.finder CreateDesktop -bool false && \
killall Finder

# Show Icons (Default)
defaults write com.apple.finder CreateDesktop -bool true && \
killall Finder
```

Path Bar

```
# Show
defaults write com.apple.finder ShowPathbar -bool true

# Hide (Default)
defaults write com.apple.finder ShowPathbar -bool false
```

Scrollbar Visibility

Possible values: `WhenScrolling`, `Automatic` and `Always`.

```
defaults write -g AppleShowScrollBars -string "Always"
```

Status Bar

```
# Show
defaults write com.apple.finder ShowStatusBar -bool true

# Hide (Default)
defaults write com.apple.finder ShowStatusBar -bool false
```

Save to Disk by Default

Sets default save target to be a local disk, not iCloud.

```
defaults write -g NSDocumentSaveNewDocumentsToCloud -bool false
```

Set Current Folder as Default Search Scope

```
defaults write com.apple.finder FXDefaultSearchScope -string "SCcf"
```

Set Default Finder Location to Home Folder

```
defaults write com.apple.finder NewWindowTarget -string "PfLo" && \
defaults write com.apple.finder NewWindowTargetPath -string "file://${HOME}"
```

Set Sidebar Icon Size

Sets size to 'medium'.

```
defaults write -g NSTableViewDefaultSizeMode -int 2
```

Metadata Files

Disable Creation of Metadata Files on Network Volumes

Avoids creation of `.DS_Store` and AppleDouble files.

```
defaults write com.apple.desktopservices DSDontWriteNetworkStores -bool true
```

Disable Creation of Metadata Files on USB Volumes

Avoids creation of `.DS_Store` and AppleDouble files.

```
defaults write com.apple.desktopservices DSDontWriteUSBStores -bool true
```

Opening Things

Change Working Directory to Finder Path

If multiple windows are open, it chooses the top-most one.

```
cd "$(osascript -e 'tell app "Finder" to POSIX path of (insertion location as alias)')"
```

Open URL

```
open https://github.com
```

Open File

```
open README.md
```

Open Applications

You can open applications using `-a`.

```
open -a "Google Chrome" https://github.com
```

Open Folder

```
open /path/to/folder/
```

Open Current Folder

```
open .
```

Fonts

Clear Font Cache for Current User

To clear font caches for all users, put `sudo` in front of this command.

```
atsutil databases -removeUser && \
atsutil server -shutdown && \
atsutil server -ping
```

Get SF Mono Fonts

You need to download and install Xcode 8 beta for this to work. Afterwards they should be available in all applications.

```
cp -v /Applications/Xcode-beta.app/Contents/SharedFrameworks/DVTKit.framework/Versions/A/Resourc
* ~/Library/Fonts
```

From Sierra onward, they are included in Terminal.app.

```
cp -v /Applications/Utilities/Terminal.app/Contents/Resources/Fonts/SFMono-* ~/Library/Fonts
```

Starting in Catalina, the Utilities apps (including Terminal.app) are now found in the `/System` folder.

```
cp -v /System/Applications/Utilities/Terminal.app/Contents/Resources/Fonts/SFMono-* ~
/Library/Fonts
```

Functions

Please see this file.

Hardware

Bluetooth

```
# Status
defaults read /Library/Preferences/com.apple.Bluetooth ControllerPowerState

# Enable (Default)
sudo defaults write /Library/Preferences/com.apple.Bluetooth ControllerPowerState -int 1

# Disable
sudo defaults write /Library/Preferences/com.apple.Bluetooth ControllerPowerState -int 0 && \
sudo killall -HUP blued
```

Harddisks

Force Enable Trim

Enable Trim for non-Apple SSDs. This command is available since Yosemite.

```
forcetrim
```

Hardware Information

List All Hardware Ports

```
networksetup -listallhardwareports
```

Remaining Battery Percentage

```
pmset -g batt | egrep "([0-9]+\%).*" -o --colour=auto | cut -f1 -d';'
```

Remaining Battery Time

```
pmset -g batt | egrep "([0-9]+\%).*" -o --colour=auto | cut -f3 -d';'
```

Show Connected Device's UDID

```
system_profiler SPUSBDataType | sed -n -e '/iPad/,/Serial/p' -e '/iPhone/,/Serial/p'
```

Show Current Screen Resolution

```
system_profiler SPDisplaysDataType | grep Resolution
```

Show CPU Brand String

```
sysctl -n machdep.cpu.brand_string
```

Infrared Receiver

```
# Status
defaults read /Library/Preferences/com.apple.driver.AppleIRController DeviceEnabled

# Enable (Default)
defaults write /Library/Preferences/com.apple.driver.AppleIRController DeviceEnabled -int 1

# Disable
defaults write /Library/Preferences/com.apple.driver.AppleIRController DeviceEnabled -int 0
```

Power Management

Prevent System Sleep

Prevent sleep for 1 hour:

```
caffeinate -u -t 3600
```

Show All Power Management Settings

```
sudo pmset -g
```

Put Display to Sleep after 15 Minutes of Inactivity

```
sudo pmset displaysleep 15
```

Put Computer to Sleep after 30 Minutes of Inactivity

```
sudo pmset sleep 30
```

Check System Sleep Idle Time

```
sudo systemsetup -getcomputersleep
```

Set System Sleep Idle Time to 60 Minutes

```
sudo systemsetup -setcomputersleep 60
```

Turn Off System Sleep Completely

```
sudo systemsetup -setcomputersleep Never
```

Automatic Restart on System Freeze

```
sudo systemsetup -setrestartfreeze on
```

Chime When Charging

Play iOS charging sound when MagSafe is connected.

```
# Enable
defaults write com.apple.PowerChime ChimeOnAllHardware -bool true && \
open /System/Library/CoreServices/PowerChime.app

# Disable (Default)
defaults write com.apple.PowerChime ChimeOnAllHardware -bool false && \
killall PowerChime
```

Input Devices

Keyboard

Auto-Correct

```
# Disable
defaults write -g NSAutomaticSpellingCorrectionEnabled -bool false

# Enable (Default)
defaults write -g NSAutomaticSpellingCorrectionEnabled -bool true

# Show Status
defaults read -g NSAutomaticSpellingCorrectionEnabled
```

Full Keyboard Access

Enable Tab in modal dialogs.

```
# Text boxes and lists only (Default)
defaults write NSGlobalDomain AppleKeyboardUIMode -int 0

# All controls
defaults write NSGlobalDomain AppleKeyboardUIMode -int 3
```

Key Repeat

Disable the default "press and hold" behavior.

```
# Enable Key Repeat
defaults write -g ApplePressAndHoldEnabled -bool false

# Disable Key Repeat
defaults write -g ApplePressAndHoldEnabled -bool true
```

Key Repeat Rate

Sets a very fast repeat rate, adjust to taste.

```
defaults write -g KeyRepeat -int 0.02
```

Launchpad

Reset Launchpad Layout

You need to restart `Dock` because Launchpad is tied to it.

```
# Up to Yosemite
rm ~/Library/Application\ Support/Dock/*.db && \
killall Dock

# From El Capitan
defaults write com.apple.dock ResetLaunchPad -bool true && \
killall Dock
```

Media

Audio

Convert Audio File to iPhone Ringtone

```
afconvert input.mp3 ringtone.m4r -f m4af
```

Create Audiobook From Text

Uses "Alex" voice, a plain UTF-8 encoded text file for input and AAC output.

```
say -v Alex -f file.txt -o "output.m4a"
```

Disable Sound Effects on Boot

```
sudo nvram SystemAudioVolume=" "
```

Mute Audio Output

```
osascript -e 'set volume output muted true'
```

Set Audio Volume

```
osascript -e 'set volume 4'
```

Play Audio File

You can play all audio formats that are natively supported by QuickTime.

```
afplay -q 1 filename.mp3
```

Speak Text with System Default Voice

```
say 'All your base are belong to us!'
```

Startup Chime

Older Macs:

```
# Enable (Default)
sudo nvram BootAudio=%01

# Disable
sudo nvram BootAudio=%00
```

From 2016 models onwards:

```
# Enable
sudo nvram StartupMute=%00

# Disable (Default)
sudo nvram StartupMute=%01
```

Video

Auto-Play Videos in QuickTime Player

```
defaults write com.apple.QuickTimePlayerX MGPlayMovieOnOpen 1
```

Networking

Bonjour

Bonjour Service

```
# Disable
sudo defaults write /System/Library/LaunchDaemons/com.apple.mDNSResponder.plist ProgramArguments
"-NoMulticastAdvertisements"

# Enable (Default)
sudo defaults write /System/Library/LaunchDaemons/com.apple.mDNSResponder.plist ProgramArguments
"/usr/sbin/mDNSResponder" "-launchd"
```

DHCP

Renew DHCP Lease

```
sudo ipconfig set en0 DHCP
```

Show DHCP Info

```
ipconfig getpacket en0
```

DNS

Clear DNS Cache

```
sudo dscacheutil -flushcache && \
sudo killall -HUP mDNSResponder
```

Hostname

Set Computer Name/Host Name

```
sudo scutil --set ComputerName "newhostname" && \
sudo scutil --set HostName "newhostname" && \
sudo scutil --set LocalHostName "newhostname" && \
sudo defaults write /Library/Preferences/SystemConfiguration/com.apple.smb.server NetBIOSName -s
"newhostname"
```

Network Preferences

Network Locations

Switch between network locations created in the Network preference pane.

```
# Status
scselect

# Switch Network Location
scselect LocationNameFromStatus
```

Set Static IP Address

```
networksetup -setmanual "Ethernet" 192.168.2.100 255.255.255.0 192.168.2.1
```

Networking Tools

Ping a Host to See Whether It’s Available

```
ping -o github.com
```

Troubleshoot Routing Problems

```
traceroute github.com
```

SSH

Permanently Add Private Key Passphrase to SSH Agent

> Prior to macOS Sierra, ssh would present a dialog asking for your passphrase and would offer the option to store it into the keychain. This UI was deprecated some time ago and has been removed.
>
> Instead, a new UseKeychain option was introduced in macOS Sierra allowing users to specify whether they would like for the passphrase to be stored in the keychain. This option was enabled by default on macOS Sierra, which caused all passphrases to be stored in the keychain.
>
> This was not the intended default behavior, so this has been changed in macOS 10.12.2. (Source)

```
ssh-add -K /path/to/private_key
```

Then add to `~/.ssh/config`:

```
Host server.example.com
    IdentityFile /path/to/private_key
    UseKeychain yes
```

Remote Login

```
# Enable remote login
sudo launchctl load -w /System/Library/LaunchDaemons/ssh.plist

# Disable remote login
sudo launchctl unload -w /System/Library/LaunchDaemons/ssh.plist
```

TCP/IP

Show Application Using a Certain Port

This outputs all applications currently using port 80.

```
sudo lsof -i :80
```

Show External IP Address

Works if your ISP doesn't replace DNS requests (which it shouldn't).

```
dig +short myip.opendns.com @resolver1.opendns.com
```

Alternative that works on all networks.

```
curl -s https://api.ipify.org && echo
```

Show Network Interface Information

Undocumented flag of the `scutil` command.

```
scutil --nwi
```

TFTP

Start Native TFTP Daemon

Files will be served from `/private/tftpboot`.

```
sudo launchctl load -F /System/Library/LaunchDaemons/tftp.plist && \
sudo launchctl start com.apple.tftpd
```

Wi-Fi

Join a Wi-Fi Network

```
networksetup -setairportnetwork en0 WIFI_SSID WIFI_PASSWORD
```

Scan Available Access Points

Create a symbolic link to the airport command for easy access:

```
sudo ln -s /System/Library/PrivateFrameworks/Apple80211.framework/Versions/Current/Resources/air
```

Run a wireless scan:

```
airport -s
```

Show Current SSID

```
/System/Library/PrivateFrameworks/Apple80211.framework/Versions/Current/Resources/airport -I |
 awk '/ SSID/ {print substr($0, index($0, $2))}'
```

Show Local IP Address

```
ipconfig getifaddr en0
```

Show Wi-Fi Connection History

```
defaults read /Library/Preferences/SystemConfiguration/com.apple.airport.preferences |
 grep LastConnected -A 7
```

Show Wi-Fi Network Passwords

Exchange SSID with the SSID of the access point you wish to query the password from.

```
security find-generic-password -D "AirPort network password" -a "SSID" -gw
```

Turn on Wi-Fi Adapter

```
networksetup -setairportpower en0 on
```

Package Managers

- Fink - The full world of Unix Open Source software for Darwin. A little outdated.
- Homebrew - The missing package manager for OS X. The most popular choice.
- MacPorts - Compile, install and upgrade either command-line, X11 or Aqua based open-source software. Very clean, it's what I use.

Homebrew

Full Uninstall

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/uninstall)"
```

Printing

Clear Print Queue

```
cancel -a -
```

Expand Print Panel by Default

```
defaults write -g PMPrintingExpandedStateForPrint -bool true && \
defaults write -g PMPrintingExpandedStateForPrint2 -bool true
```

Quit Printer App After Print Jobs Complete

```
defaults write com.apple.print.PrintingPrefs "Quit When Finished" -bool true
```

Security

Application Firewall

Firewall Service

```
# Show Status
sudo /usr/libexec/ApplicationFirewall/socketfilterfw --getglobalstate

# Enable
sudo /usr/libexec/ApplicationFirewall/socketfilterfw --setglobalstate on
```

```
# Disable (Default)
sudo /usr/libexec/ApplicationFirewall/socketfilterfw --setglobalstate off
```

Add Application to Firewall

```
sudo /usr/libexec/ApplicationFirewall/socketfilterfw --add /path/to/file
```

Gatekeeper

Add Gatekeeper Exception

```
spctl --add /path/to/Application.app
```

Remove Gatekeeper Exception

```
spctl --remove /path/to/Application.app
```

Manage Gatekeeper

Especially helpful with the annoying Catalina system popup blocking execution of non-signed apps.

```
# Status
spctl --status

# Enable (Default)
sudo spctl --master-enable

# Disable
sudo spctl --master-disable
```

Passwords

Generate Secure Password and Copy to Clipboard

```
LC_ALL=C tr -dc "[:alnum:]" < /dev/urandom | head -c 20 | pbcopy
```

Physical Access

Launch Screen Saver

```
# Up to Sierra
open /System/Library/Frameworks/ScreenSaver.framework/Versions/A/Resources/ScreenSaverEngine.app

# From High Sierra
```

```
/System/Library/CoreServices/ScreenSaverEngine.app/Contents/MacOS/ScreenSaverEngine
```

Lock Screen

```
/System/Library/CoreServices/Menu\ Extras/User.menu/Contents/Resources/CGSession -suspend
```

Screensaver Immediate Lock

```
# Status
defaults read com.apple.screensaver askForPasswordDelay

# Enable (Default)
defaults write com.apple.screensaver askForPasswordDelay -int 0

# Disable (Integer = lock delay in seconds)
defaults write com.apple.screensaver askForPasswordDelay -int 10
```

Screensaver Password

```
# Status
defaults read com.apple.screensaver askForPassword

# Enable
defaults write com.apple.screensaver askForPassword -int 1

# Disable (Default)
defaults write com.apple.screensaver askForPassword -int 0
```

Wiping Data

Note: The `srm` command appears to have been removed on MacOS after 10.9. There is a note on an Apple support page hinting as to why:

> With an SSD drive, Secure Erase and Erasing Free Space are not available in Disk Utility. These options are not needed for an SSD drive because a standard erase makes it difficult to recover data from an SSD.

Securely Remove File

```
srm /path/to/file
```

Securely Remove Folder

```
srm -r /path/to/folder/
```

Securely Remove Path (Force)

```
srm -rf /path/to/complete/destruction
```

Search

Find

Recursively Delete .DS_Store Files

```
find . -type f -name '*.DS_Store' -ls -delete
```

Locate

Build Locate Database

```
sudo launchctl load -w /System/Library/LaunchDaemons/com.apple.locate.plist
```

Search via Locate

The `-i` modifier makes the search case insensitive.

```
locate -i *.jpg
```

System

AirDrop

```
# Enable AirDrop over Ethernet and on Unsupported Macs
defaults write com.apple.NetworkBrowser BrowseAllInterfaces -bool true

# Enable (Default)
defaults remove com.apple.NetworkBrowser DisableAirDrop

# Disable
defaults write com.apple.NetworkBrowser DisableAirDrop -bool YES
```

AppleScript

Execute AppleScript

```
osascript /path/to/script.scpt
```

Basics

Compare Two Folders

```
diff -qr /path/to/folder1 /path/to/folder2
```

Copy Large File with Progress

Make sure you have `pv` installed and replace `/dev/rdisk2` with the appropriate write device or file.

```
FILE=/path/to/file.iso pv -s $(du -h $FILE | awk '/.*/ {print $1}') $FILE |
 sudo dd of=/dev/rdisk2 bs=1m
```

Restore Sane Shell

In case your shell session went insane (some script or application turned it into a garbled mess).

```
stty sane
```

Restart

```
sudo reboot
```

Shutdown

```
sudo poweroff
```

Show Build Number of OS

```
sw_vers
```

Uptime

How long since your last restart.

```
uptime
```

Clipboard

Copy data to Clipboard

```
cat whatever.txt | pbcopy
```

Convert Clipboard to Plain Text

```
pbpaste | textutil -convert txt -stdin -stdout -encoding 30 | pbcopy
```

Convert Tabs to Spaces for Clipboard Content

```
pbpaste | expand | pbcopy
```

Copy data from Clipboard

```
pbpaste > whatever.txt
```

Sort and Strip Duplicate Lines from Clipboard Content

```
pbpaste | sort | uniq | pbcopy
```

FileVault

Automatically Unlock FileVault on Restart

If FileVault is enabled on the current volume, it restarts the system, bypassing the initial unlock. The command may not work on all systems.

```
sudo fdesetup authrestart
```

FileVault Service

```
# Status
sudo fdesetup status

# Enable
sudo fdesetup enable

# Disable (Default)
sudo fdesetup disable
```

Information/Reports

Generate Advanced System and Performance Report

```
sudo sysdiagnose -f ~/Desktop/
```

Install OS

Create Bootable Installer

```
# Mojave
sudo /Applications/Install\ macOS\
Mojave.app/Contents/Resources/createinstallmedia --volume /Volumes/USB --nointeraction --downloa

# High Sierra
sudo /Applications/Install\ macOS\ High\
Sierra.app/Contents/Resources/createinstallmedia --volume /Volumes/USB --applicationpath /Applic
\ macOS\ High\ Sierra.app

# Sierra
sudo /Applications/Install\ macOS\
Sierra.app/Contents/Resources/createinstallmedia --volume /Volumes/USB --applicationpath /Applic
\ macOS\ Sierra.app

# El Capitan
sudo /Applications/Install\ OS\ X\ El\
Capitan.app/Contents/Resources/createinstallmedia --volume /Volumes/USB --applicationpath /Appli
\ OS\ X\ El\ Capitan.app

# Yosemite
sudo /Applications/Install\ OS\ X\
Yosemite.app/Contents/Resources/createinstallmedia --volume /Volumes/USB --applicationpath /Appl
\ OS\ X\ Yosemite.app
```

- For confirmation before erasing the drive, remove `—-nointeraction` from the command.
- The optional `—-downloadassets` flag is new in Mojave. It downloads assets which may be required during installation, like updates.
- The `—-applicationpath` flag is deprecated since Mojave and will throw an error if used.

Kernel Extensions

Display Status of Loaded Kernel Extensions

```
sudo kextstat -l
```

Load Kernel Extension

```
sudo kextload -b com.apple.driver.ExampleBundle
```

Unload Kernel Extensions

```
sudo kextunload -b com.apple.driver.ExampleBundle
```

LaunchAgents

Please see this file.

LaunchServices

Rebuild LaunchServices Database

To be independent of OS X version, this relies on `locate` to find `lsregister`. If you do not have your `locate` database built yet, do it.

```
sudo $(locate lsregister) -kill -seed -r
```

Login Window

Set Login Window Text

```
sudo defaults write /Library/Preferences/com.apple.loginwindow LoginwindowText "Your text"
```

Memory Management

Purge memory cache

```
sudo purge
```

Show Memory Statistics

```
# One time
vm_stat

# Table of data, repeat 10 times total, 1 second wait between each poll
vm_stat -c 10 1
```

Notification Center

Notification Center Service

```
# Disable
launchctl unload -w /System/Library/LaunchAgents/com.apple.notificationcenterui.plist && \
killall -9 NotificationCenter

# Enable (Default)
```

```
launchctl load -w /System/Library/LaunchAgents/com.apple.notificationcenterui.plist
```

QuickLook

Preview via QuickLook

```
qlmanage -p /path/to/file
```

Remote Apple Events

```
# Status
sudo systemsetup -getremoteappleevents

# Enable
sudo systemsetup -setremoteappleevents on

# Disable (Default)
sudo systemsetup -setremoteappleevents off
```

Root User

```
# Enable
dsenableroot

# Disable
dsenableroot -d
```

Safe Mode Boot

```
# Status
nvram boot-args

# Enable
sudo nvram boot-args="-x"

# Disable
sudo nvram boot-args=""
```

Save Dialogs

Significantly improve the now rather slow animation in save dialogs.

```
defaults write NSGlobalDomain NSWindowResizeTime .001
```

Screenshots

Take Delayed Screenshot

Takes a screenshot as JPEG after 3 seconds and displays in Preview.

```
screencapture -T 3 -t jpg -P delayedpic.jpg
```

Save Screenshots to Given Location

Sets location to `~/Desktop`.

```
defaults write com.apple.screencapture location ~/Desktop && \
killall SystemUIServer
```

Save Screenshots in Given Format

Sets format to `png`. Other options are `bmp`, `gif`, `jpg`, `jpeg`, `pdf`, `tiff`.

```
defaults write com.apple.screencapture type -string "png"
```

Disable Shadow in Screenshots

```
defaults write com.apple.screencapture disable-shadow -bool true && \
killall SystemUIServer
```

Set Default Screenshot Name

Date and time remain unchanged.

```
defaults write com.apple.screencapture name "Example name" && \
killall SystemUIServer
```

Software Installation

Install PKG

```
installer -pkg /path/to/installer.pkg -target /
```

Sidecar

Use on Incompatible Macs

This may or may not work, depending on the age of the machine.

```
# Enable
defaults write com.apple.sidecar.display AllowAllDevices -bool true && \
defaults write com.apple.sidecar.display hasShownPref -bool true

# Disable (Default)
defaults delete com.apple.sidecar.display
```

Software Update

Ignore Specific Software Update

The identifier can be found via `softwareupdate --list`. In the example below, being on Mojave, will ignore all update prompts to Catalina, since the latter removes 32-bit support.

```
sudo /usr/sbin/softwareupdate --ignore "macOS Catalina"
```

Install All Available Software Updates

```
sudo softwareupdate -ia
```

Set Software Update Check Interval

Set to check daily instead of weekly.

```
defaults write com.apple.SoftwareUpdate ScheduleFrequency -int 1
```

Show Available Software Updates

```
sudo softwareupdate --list
```

Set Software Update Server

This should only be done for testing purposes or unmanaged clients. To use network-wide, either correctly set up DNS along with Apple SUS service and bind your clients via OpenDirectory. Alternatively, use Reposado together with correct network DNS settings to make resolution transparent. Margarita looks nice to have as well.

```
# Use own SUS
sudo defaults write /Library/Preferences/com.apple.SoftwareUpdate CatalogURL http://su.example.c

# Reset to Apple SUS
sudo defaults delete /Library/Preferences/com.apple.SoftwareUpdate CatalogURL
```

Software Version

Show System Software Version

```
sw_vers -productVersion
```

Spotlight

Spotlight Indexing

```
# Disable
mdutil -i off -d /path/to/volume

# Enable (Default)
mdutil -i on /path/to/volume
```

Erase Spotlight Index and Rebuild

```
mdutil -E /path/to/volume
```

Search via Spotlight

```
mdfind -name 'searchterm'
```

Show Spotlight Indexed Metadata

```
mdls /path/to/file
```

System Integrity Protection

Disable System Integrity Protection

Reboot while holding `Cmd` + `R`, open the Terminal application and enter:

```
csrutil disable && reboot
```

Enable System Integrity Protection

Reboot while holding `Cmd` + `R`, open the Terminal application and enter:

```
csrutil enable && reboot
```

Date and Time

List Available Timezones

```
sudo systemsetup -listtimezones
```

Set Timezone

```
sudo systemsetup -settimezone Europe/Berlin
```

Set Clock Using Network Time

```
# Status
sudo systemsetup getusingnetworktime

# Enable (Default)
sudo systemsetup setusingnetworktime on

# Disable
sudo systemsetup setusingnetworktime off
```

Terminal

Ring Terminal Bell

Rings the terminal bell (if enabled) and puts a badge on it.

```
tput bel
```

Alternative Terminals

- Alacritty - Cross-platform, GPU-accelerated terminal emulator.
- iTerm2 - A better Terminal.app.
- kitty - Modern, GPU-accelerated terminal emulator.

Shells

Bash

Install the latest version and set as current user's default shell:

```
brew install bash && \
echo $(brew --prefix)/bin/bash | sudo tee -a /etc/shells && \
chsh -s $(brew --prefix)/bin/bash
```

- Homepage - The default shell for OS X and most other Unix-based operating systems.
- Bash-it - Community Bash framework, like Oh My Zsh for Bash.

fish

Install the latest version and set as current user's default shell:

```
brew install fish && \
echo $(brew --prefix)/bin/fish | sudo tee -a /etc/shells && \
chsh -s $(brew --prefix)/bin/fish
```

- Homepage - A smart and user-friendly command line shell for OS X, Linux, and the rest of the family.
- The Fishshell Framework - Provides core infrastructure to allow you to install packages which extend or modify the look of your shell.
- Installation & Configuration Tutorial - How to Setup Fish Shell with Fisherman, Powerline Fonts, iTerm2 and Budspencer Theme on OS X.

Zsh

Install the latest version and set as current user's default shell:

```
brew install zsh && \
sudo sh -c 'echo $(brew --prefix)/bin/zsh >> /etc/shells' && \
chsh -s $(brew --prefix)/bin/zsh
```

- Homepage - Zsh is a shell designed for interactive use, although it is also a powerful scripting language.
- Oh My Zsh - An open source, community-driven framework for managing your Zsh configuration.
- Prezto - A speedy Zsh framework. Enriches the command line interface environment with sane defaults, aliases, functions, auto completion, and prompt themes.
- zgen - Another open source framework for managing your zsh configuration. Zgen will load oh-my-zsh compatible plugins and themes and has the advantage of both being faster and automatically cloning any plugins used in your configuration for you.

Terminal Fonts

- Anonymous Pro - A family of four fixed-width fonts designed with coding in mind.
- Codeface - A gallery and repository of monospaced fonts for developers.
- DejaVu Sans Mono - A font family based on the Vera Fonts.
- Hack - Hack is hand groomed and optically balanced to be your go-to code face.
- Inconsolata - A monospace font, designed for code listings and the like.
- Input - A flexible system of fonts designed specifically for code.
- Meslo - Customized version of Apple's Menlo font.
- Operator Mono - A surprisingly usable alternative take on a monospace font (commercial).
- Powerline Fonts - Repo of patched fonts for the Powerline plugin.
- Source Code Pro - A monospaced font family for user interfaces and coding environments.

Glossary

Mac OS X, OS X, and macOS Version Information

Version	Codename	Release Date	Most Recent Version
Rhapsody Developer Release	Grail1Z4 / Titan1U	August 31, 1997	DR2 (May 14, 1998)
Mac OS X Server 1.0	Hera	March 16, 1999	1.2v3 (October 27, 2000)
Mac OS X Developer Preview	n/a	March 16, 1999	DP4 (April 5, 2000)
Mac OS X Public Beta	Kodiak	September 13, 2000	n/a
Mac OS X 10.0	Cheetah	March 24, 2001	10.0.4 (June 22, 2001)
Mac OS X 10.1	Puma	September 25, 2001	10.1.5 (June 6, 2002)
Mac OS X 10.2	Jaguar	August 24, 2002	10.2.8 (October 3, 2003)
Mac OS X 10.3	Panther	October 24, 2003	10.3.9 (April 15, 2005)
Mac OS X 10.4	Tiger	April 29, 2005	10.4.11 (November 14, 2007)
Mac OS X 10.5	Leopard	October 26, 2007	10.5.8 (August 5, 2009)
Mac OS X 10.6	Snow Leopard	August 28, 2009	10.6.8 v1.1 (July 25, 2011)
Mac OS X 10.7	Lion	July 20, 2011	10.7.5 (September 19, 2012)
OS X 10.8	Mountain Lion	July 25, 2012	10.8.5 (12F45) (October 3, 2013)
OS X 10.9	Mavericks	October 22, 2013	10.9.5 (13F1112) (September 18, 2014)
OS X 10.10	Yosemite	October 16, 2014	10.10.5 (14F27) (August 13, 2015)
OS X 10.11	El Capitan	September 30, 2015	10.11.6 (15G31) (July 18, 2016)
macOS 10.12	Sierra	September 20, 2016	10.12.6 (16G29) (July 19, 2017)
macOS 10.13	High Sierra	September 25, 2017	10.13.6 (17G65) (July 9, 2018)
macOS 10.14	Mojave	September 24, 2018	10.14.6 (18G3020) (January 28, 2020)
macOS 10.15	Catalina	October 7, 2019	10.15.3 (19D76) (January 28, 2020)

License

Mac OsX

Créer une clé USB bootable Osx

Introduction

Créer une clé USB bootable pour Mac OSX El Capitan, Yosemite, Mavericks, macOS Sierra, macOS High Sierra, macOS Mojave, macOS Catalina et macOS Big Sur.

(Le processus est également décrit sur la page de support Apple à l'adresse https://support.apple.com/fr-fr/HT201372.)

- ## **Étape 1** Créer une clé USB bootable

  - Téléchargez le fichier d'installation dans le Mac App Store
  - Si vous voulez télécharger d'anciens fichiers d'installation que vous avez téléchargés, allez sur l'onglet **"Achats"**.
  - Il est impossible de télécharger les versions antérieures de macOS ou Mac OSX sur le Mac App Store depuis la version macOS Mojave.

Étape 2

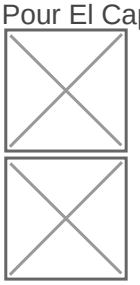
Insérez votre clé USB dans le port USB de votre Mac.

- Insérez votre **clé USB** dans le port USB de votre Mac.
- **Attention** : votre clé va être formatée plus tard, veillez à ne pas avoir de données importantes dessus.
  Faites une recherche avec le Spotlight ou allez dans "Applications/Utilitaire" pour lancer le T
  Cliquez dessus pour lancer le Terminal.Cliquez dessus pour lancer le Terminal.

- Faites une recherche avec le **Spotlight** ou allez dans **"Applications/Utilitaire"** pour lancer le **Terminal**.
- Cliquez dessus pour lancer le **Terminal**.

Étape 3

Dans le Terminal, insérez une des ligne de commande suivantes (Le nom du système d'exploitation est
Pour El Capitan : sudo /Applications/Install\ OS\ X\ El\ Capitan.app/Contents/Resources/createinstallme



- Dans le **Terminal**, insérez une des ligne de commande suivantes (Le nom du système d'exploitation est celui que vous voulez installer sur la clé. Pas celui du système de votre Mac) :
- **Pour El Capitan :** `sudo /Applications/Install\ OS\ X\ El\ Capitan.app/Contents/Resources/createinstallmedia --volume /Volumes/MyVolume --applicationpath /Applications/Install\ OS\ X\ El\ Capitan.app`
- **Pour Yosemite :** `sudo /Applications/Install\ OS\ X\ Yosemite.app/Contents/Resources/createinstallmedia --volume /Volumes/MyVolume --applicationpath /Applications/Install\ OS\ X\ Yosemite.app`
- **Pour Mavericks :** `sudo /Applications/Install\ OS\ X\ Mavericks.app/Contents/Resources/createinstallmedia --volume /Volumes/MyVolume --applicationpath /Applications/Install\ OS\ X\ Mavericks.app`
- **Pour Sierra :** `sudo /Applications/Install\ macOS\ Sierra.app/Contents/Resources/createinstallmedia --volume /Volumes/MyVolume --applicationpath /Applications/Install\ macOS\ Sierra.app`
- **Suite à l'étape suivante.**
- Remplacer le chemin **"/Volumes/MyVolume"** par celui de votre clé USB, Pour ce faire, supprimez le chemin actuel puis faites un glisser-déposer.
-

Ubuntu on a MacBookPro 8,2

Partition

In OSX, open the Disk Utility, select the main disk. We're going to split this in to two parts. Up to you how much you want to dedicate to OSX. Set the 2nd partition to be "Free Space".

rEFInd

Because Apple is a super-special-snowflake, it's not possible to directly install Ubuntu onto the new partition. We need to first install rEFInd. There are specific instructions for Yosemite.

1. Download rEFInd.
2. Extract it onto your Mac.
3. Open a Terminal and navigate to the directory.
4. Run this command

   ```
   sudo ./install.sh --esp
   ```

5. We need to edit some files. Still in the Terminal

   ```
   mkdir /Volumes/esp
   ```

   then

   ```
   sudo mount -t msdos /dev/disk0s1 /Volumes/esp
   ```

6. Edit the file "/Volumes/esp/EFI/refind/refind.conf". Change the line

   ```
   #dont_scan_volumes
   ```

   to

   ```
   dont_scan_volumes foo,bar
   ```

7. Save the file. When you reboot, rEFInd should ask you which OS you want to boot to.

Get Ubuntu

Probably the easiest bit! Download the 64 bit version of Ubuntu. I used another Ubuntu machine to create the Boot Disk on a USB stick.

Install Ubuntu

Turn your MacBook off, insert the USB drive, turn the device on. All being well, rEFInd will give you a choice of devices to boot into. You *probably* want the one which says something like

```
/USB/EFI/grub64
```

You should see a grub style boot screen. Use the cursor keys to move down to "Install Ubuntu". **Do NOT press enter!** Press "e" to edit the boot options.

If you don't do this - your screen will be blank. And a black screen will make you sad.

Remove the words

```
quiet nosplash
```

and replace them with

```
nomodeset
```

. Then press the "F10" key to boot.

You'll be taken through the normal Ubuntu installation screen. It's pretty much just clicking "Next" until it is installed.

Partitions

...With the exception of setting up the partitions!

- Create a 130MB empty partion. Macs get sad if OSX is bundled up next to a proper OS.
- Create a 512MB partition and assign it to "/boot"
- At the end of the free space, create a "Swap" parition. Personally, I use as much swap as there is RAM.
- With the remaining free space, create a partition and assign it to "/".
- Which device to you want to boot from? I chose the partition assigned to "/boot" - that way you go straight into Ubuntu when you start the machine. If you ever need to get into OSX, hold down "alt" while turning the machine on.

Keep clicking next! Eventually Ubuntu will be installed and you can reboot!

Setting Up Video

When the grub screen comes up, again you'll need to hit "e" to edit the default boot options.

Scroll down to "load_video" and add these lines below it. I don't know what they mean - it's just magic.

Lorsque l'écran grub apparaît, vous devrez à nouveau appuyer sur "e" pour modifier les options de démarrage par défaut.

```
 outb 0x728 1
 outb 0x710 2
 outb 0x740 2
 outb 0x750 0
```

```
set timeout=5
menuentry "Try Ubuntu without installing" {
   set gfxpayload=keep
   outb 0x728 1
   outb 0x710 2
   outb 0x740 2
   outb 0x750 0
   linux   /casper/vmlinuz  file=/cdrom/preseed/ubuntu.seed boot=casper quiet splash
i915.lvds_channel_mode=2 i915.modeset=1 i915.lvds_use_ssc=0 ---
   initrd   /casper/initrd
}
```

Faites défiler jusqu'à "load_video" et ajoutez ces lignes en dessous. Je ne sais pas ce qu'ils veulent dire, c'est juste magique .

Cela devrait vous démarrer sous Linux. Nous devons maintenant rendre ces changements permanents. Ouvrez un terminal et exécutez

```
sudo nano /etc/grub.d/10_linux
```

Appuyez sur CTRL+W pour trouver la ligne contenant "gzio".

Ajoutez ces lignes avant, de sorte que la nouvelle section ressemble à

```
echo " outb 0x728 1" | sed "s/^/$submenu_indentation/"
echo " outb 0x710 2" | sed "s/^/$submenu_indentation/"
echo " outb 0x740 2" | sed "s/^/$submenu_indentation/"
echo " outb 0x750 0" | sed "s/^/$submenu_indentation/"
echo " insmod gzio" | sed "s/^/$submenu_indentation/"
```

Pour appliquer ces modifications, exécutez les commandes suivantes.

```
sudo update-grub
sudo apt-get update
sudo apt-get dist-upgrade
```

Ventilateurs

Pour une meilleure performance du ventilateur...

```
sudo apt-get install lm-sensors
sudo sensor-detect
```

Répondez oui à toutes les questions. Puis:

```
sudo apt-get install macfanctld
```

Redémarrez et profitez d'Ubuntu ! Tout semble "fonctionner" - même si j'écrirai un autre article de blog expliquant comment je l'ai personnalisé.

Wifi

```
apt-get install firmware-b43-installer
```

Chroot Grub

Boot with your Live CD, selecting "Try Ubuntu without installing".

Once it boots, open a terminal (ctrl-alt-t) and mount your Ubuntu partition on /mnt. I'm assuming the Ubuntu partition is /dev/sda5, but you should determine this yourself. Let me know if you need help to do this:

```
sudo mount /dev/sda5 /mnt
```

Then mount a few more directories that are needed:

```
sudo mount --bind /dev /mnt/dev
sudo mount --bind /sys /mnt/sys
sudo mount --bind /proc /mnt/proc
```

Also, if you have a separate Ubuntu boot partition (pretty uncommon these days, but it may be the case):

```
sudo mount /dev/sdaX /mnt/boot
```

How can you tell if you have a boot partition?

Once you have your Ubuntu partition mounted, open `/mnt/etc/fstab`. If you see an entry for `/boot`, note which device it is pointing to (`/dev/sda4` maybe?). This is the one you have to mount.

Once these are mounted, do chroot to start using the mounted directory as the root partition:

```
sudo chroot /mnt
```

You'll get a `#/` prompt. First thing to do is confirm that you're using the correct `/boot` directory. Go to `/boot/grub` and look at the files there. There should be a bunch of .mod files and a grub.cfg file. If the directory is empty, don't continue, because it means this is NOT your actual `boot` directory. Look above to see how to determine if you need to mount an additional `boot` directory.

Once you've confirmed that `/boot/` contains the correct files, meaning that it *is* the correct location, type:

```
sudo update-grub
```

```
sudo grub-install --boot-directory=/mnt/boot /dev/sda
```

This should rebuild your /boot/grub/grub.cfg file with the menu entries.

Then exit the chroot:

```
exit
```

At this point you may want to check that things were correctly updated. For this, `cd /mnt/boot/grub` and check that grub's files are there, there should be a bunch of .mod files and grub.cfg, the latter should have entries for your Ubuntu kernels. If you only see grub.cfg and no .mod files, it means that this is NOT the correct boot directory, look above for how to mount a separate boot partition.

Unmount the filesystems:

```
sudo umount /mnt/dev
sudo umount /mnt/sys
sudo umount /mnt/proc
sudo umount /mnt/boot #Only if you mounted it earlier
sudo umount /mnt/
```

And then reboot, hopefully your Grub menu will be restored.

V2

1. Install linux to a flash drive on a different PC. Load from one stick, and use another as a target drive.
2. Boot from that linux-on-stick on PC.
3. Use this tutorial https://www.variadic.xyz/2020/06/15/ubuntu-2011mbp/ to modify default GRUB options
4. Boot MBP off this modified live stick, install linux to your Mac. (In Ubuntu, the GUI for installing os is named Ubiquity, and can be installed like any other application: `apt install ubiquity`.)
5. After installation is complete, don't reboot, but chroot to an installed Ubuntu. To do that, from terminal enter the following:

```
sudo mount /dev/[your new root partition] /mnt
sudo cp /etc/resolv.conf /mnt/etc/resolv.conf
sudo mount --bind /dev /mnt/dev
sudo mount --bind /proc /mnt/proc
sudo mount --bind /sys  /mnt/sys
sudo chroot /mnt /bin/bash
sudo mount -a
```

After that, repeat step 3.

6. Exit the chroot and reboot:

```
exit
umount /mnt*
reboot
```

7. Now Macbook should boot, function with Intel GPU and survive OS updates. If not, just boot that "rescue usb" you made in step 3 and repeat steps 5-6.

Qemu for Mac M1

installation de Qemu :

```
brew install knazarov/qemu-virgl/qemu-virgl
```

Création du dossier et téléchargement de l'archive pour ArchLinux :

```
mkdir
archlinuxKDE

cd archlinuxKDE
curl -L https://github.com/m-bers/Arch-Linux-Arm-M1/releases/latest/download/archlinux.tar.gz
| tar xzf -
```

Lancement de la machine Qemu pour l'installation :

```
sudo qemu-system-aarch64 -L ~/bin/qemu/share/qemu
\
        -smp 8 \
        -machine virt,accel=hvf,highmem=off \
        -cpu cortex-a72 -m 4096 \
        -drive "if=pflash,media=disk,id=drive0,file=flash0.img,cache=writethrough,format=raw"
\
        -drive "if=pflash,media=disk,id=drive1,file=flash1.img,cache=writethrough,format=raw"
```

```
\
        -drive
"if=virtio,media=disk,id=drive2,file=archlinux.qcow2,cache=writethrough,format=qcow2" \
        -device virtio-net-pci,netdev=net \
        -netdev user,id=net,ipv6=off \
        -device virtio-rng-device -device virtio-balloon-device -device virtio-keyboard-device
\
        -device virtio-mouse-device -device virtio-serial-device -device virtio-tablet-device
\
        -object cryptodev-backend-builtin,id=cryptodev0 \
        -device virtio-crypto-pci,id=crypto0,cryptodev=cryptodev0 \
        -nographic
```

Dans la machine virtuelle :

se connecter en root :

> user : root:root

```
pacman-key --init
pacman-key --populate archlinuxarm
pacman -Suy
pacman -S nano git
```

editer le fichier pacman.conf et modifier :

Color
ParallelDownloads = 8
Modifier la langue :
nano /etc/locale.gen
locale-gen
echo “LANG=fr_FR.UTF-8” >> /etc/locale.conf

Installation de endeavouros :

```
git clone https://github.com/endeavouros-arm/install-script.git
cd install-script
chmod 777 endeavour-ARM-install-V2.6.sh
./endeavour-ARM-install-V2.6.sh
```

```
nano run.sh
==== ajouter ceci ====
sudo /opt/homebrew/Cellar/qemu/6.2.0_1/bin/qemu-system-aarch64 \
-smp 8 \
-machine virt,accel=hvf,highmem=off \
-cpu cortex-a72 -m 4096 \
-drive "if=pflash,media=disk,id=drive0,file=flash0.img,cache=writethrough,format=raw" \
-drive "if=pflash,media=disk,id=drive1,file=flash1.img,cache=writethrough,format=raw" \
-drive "if=virtio,media=disk,id=drive2,file=archlinux.qcow2,cache=writethrough,format=qcow2" \
    -device virtio-net-pci,netdev=net \
    -netdev user,id=net,ipv6=off \
    -device intel-hda -device hda-output \
    -device qemu-xhci \
    -device usb-kbd \
    -device virtio-mouse-pci \
    -device virtio-tablet-pci \
-object cryptodev-backend-builtin,id=cryptodev0 \
-device virtio-crypto-pci,id=crypto0,cryptodev=cryptodev0 \
    -display cocoa,show-cursor=on  \
 -device virtio-gpu-pci

  ======================

chmod +x run.sh
```

DVDrip HandBrake Mac M1

J'ai installé libdvdcss avec brew sur mon MacBook M1, puis j'ai lié symboliquement les bibliothèques statiques et dynamiques dans `/usr/local/lib.`

Cela n'a pas fonctionné avec le frein à main pour une raison quelconque.

Donc, à la place, j'ai juste compilé des bibliothèques à partir de la source comme ceci :

1. Download official source code (I used version 1.4.3).

```
wget http://download.videolan.org/pub/libdvdcss/1.4.3/libdvdcss-1.4.3.tar.bz2
tar -xjf ./libdvdcss-1.4.3.tar.bz2
cd libdvdcss-1.4.3
./configure --prefix=/usr/local
make
sudo make install
```

Worked fine with handbrake 1.4.2 (2021100300). In /usr/local/lib I now have this: