Bitácora Linux Mint Tara 19.1

Sergio Alvariño salvari@gmail.com

abril-2019

Resumen

Bitácora de mi portatil
Solo para referencia rápida y personal.

Índice general

1 Introducción

Mi portátil es un ordenador Acer 5755G con las siguientes características:

- Core i5 2430M 2.4GHz
- NVIDIA Geforce GT 540M
- 8Gb RAM
- 750Gb HD

Mi portátil equipa una tarjeta *Nvidia Geforce GT540M* que resulta pertenecer a una rama muerta en el árbol de desarrollo de Nvidia.

Esta tarjeta provoca todo tipo de problemas de sobrecalientamiento, no importa que versión de Linux uses.

2 Programas básicos

2.1 Linux Mint

Linux Mint incluye `sudo` [1] y las aplicaciones que uso habitualmente para gestión de paquetes por defecto (*aptitude* y *synaptic*).

Tampoco voy a enredar nada con los orígenes del software (de momento)

2.2 Firmware

Instalamos el paquete `intel-microcode` desde el gestor de drivers.

[1] ya no incluye gksu pero tampoco es imprescindible

Instalamos el driver recomendado de nvidia desde el gestor de drivers del *Linux Mint*. Ahora mismo es el *nvidia-driver-390*

Configuramos desde el interfaz del driver para activar la tarjeta intel.

Como a pesar de eso seguimos teniendo problemas de calentamiento:

```
apt install tlp
tlp start
apt install lm-sensors hddtemp
apt install linux-tools-common linux-tools-generic
cpupower frequency-set -g powersave
apt install cpufrequtils
```

Referencias:

- https://itsfoss.com/reduce-overheating-laptops-linux/
- http://www.webupd8.org/2014/04/prevent-your-laptop-from-overheating.html

Después de un reinicio **frio** [2] todo parece funcionar de nuevo.

2.3 Parámetros de disco duro

Tengo un disco duro ssd.

Añadimos el parámetro `noatime` para las particiones de `root` y `/home`.

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda5 during installation
UUID=d96a5501-75b9-4a25-8ecb-c84cd4a3fff5 /          ext4   noatime,errors=remount-
ro 0       1
# /home was on /dev/sda7 during installation
UUID=8fcde9c5-d694-4417-adc0-8dc229299f4c /home       ext4   defaults,noatime     0
# /store was on /dev/sdc7 during installation
UUID=0f0892e0-9183-48bd-aab4-9014dc1bd03a /store      ext4   defaults     0     2
```

[2] puede que haya un *bug* que hace fallar el sensor de temperatura si el portatil no arranca frio

```
# swap was on /dev/sda6 during installation
UUID=ce11ccb0-a67d-4e8b-9456-f49a52974160 none        swap   sw         0     0
# swap was on /dev/sdc5 during installation
UUID=11090d84-ce98-40e2-b7be-dce3f841d7b4 none        swap   sw         0     0
```

Una vez modificado el `/etc/fstab` no hace falta arrancar:

```
mount -o remount /
mount -o remount /home
mount
```

En el printado de `mount` ya veremos si ha cargado el parámetro.

Pasamos el `fstrim` desde weekly a daily.

Seguimos instrucciones de aquí.

Más concretamente de aquí

y cambiamos el parámetro de *swapiness* a 1.

2.4 Fuentes adicionales

Instalamos algunas fuentes desde los orígenes de software:

```
sudo apt install ttf-mscorefonts-installer
sudo apt install fonts-noto
```

Y la fuente Mensch la bajamos directamente al directorio `~/.local/share/fonts`

2.5 Firewall

`ufw` y `gufw` vienen instalados por defecto, pero no activados.

```
aptitude install ufw
ufw default deny
ufw enable
ufw status verbose
aptitude install gufw
```

2.6 Control de configuraciones con git

2.6.1 Instalación de `etckeeper`

```
sudo su -
```

```
git config --global user.email xxxxx@whatever.com
git config --global user.name "Name Surname"
apt install etckeeper
```

etckeeper hara un control automático de tus ficheros de configuración en `/etc`

Para echar una mirada a los *commits* creados puedes ejecutar:

```
cd /etc
sudo git log
```

2.6.2 Controlar dotfiles con git

Vamos a crear un repo de git para controlar nuestros ficheros personales de configuración.

Creamos el repo donde queramos

```
mkdir usrcfg
cd usrcfg
git init
git config core.worktree "/home/salvari"
```

Y ya lo tenemos, un repo que tiene el directorio de trabajo apuntando a nuestro *$HOME*.

Podemos añadir los ficheros de configuración que queramos al repo:

```
git add .bashrc
git add .zshrc
git commit -m "Add some dotfiles"
```

Una vez que he añadido los ficheros que quiero tener controlados he puesto un `*` en el fichero `.git/info/exclude` de mi repo para que ignore todos los ficheros de mi `$HOME`.

Cuando instalo algún programa nuevo añado a mano los ficheros de configuración al repo.

2.7 Aplicaciones variadas

Nota: Ya no instalamos *menulibre*, Linux Mint tiene una utilidad de edición de menús.

Keepass2 Para mantener nuestras contraseñas a buen recaudo

Gnucash Programa de contabilidad

Deluge Programa de descarga de torrents (acuérdate de configurar tus cortafuegos)

Chromium Como Chrome pero libre

rsync, grsync Para hacer backups de nuestros ficheros

Descompresores variados Para lidiar con los distintos formatos de ficheros comprimidos

```
sudo apt install keepass2 gnucash deluge rsync grsync rar unrar \
zip unzip unace bzip2 lzop p7zip p7zip-full p7zip-rar chromium-browser
```

2.8 Programas de terminal

Dos imprescindibles:

```
sudo apt install guake terminator
```

TODO: asociar *Guake* a una combinación apropiada de teclas.

2.9 Dropbox

Lo instalamos desde el software manager.

2.10 Chrome

Instalado desde la página web de Chrome

2.11 Varias aplicaciones instaladas de binarios

Lo recomendable en un sistema POSIX es instalar los programas adicionales en `/usr/local` o en `/opt`. Yo soy más chapuzas y suelo instalar en `~/apt` por que el portátil es personal e intrasferible. En un ordenador compartido es mejor usar `/opt`.

2.11.1 Freeplane

Para hacer mapas mentales, presentaciones, resúmenes, apuntes… La versión incluida en LinuxMint está un poco anticuada.

1. descargamos desde la web.
2. Descomprimimos en `~/apps/freeplane`
3. Creamos enlace simbólico
4. Añadimos a los menús

2.11.2 Telegram Desktop

Cliente de Telegram, descargado desde la página web.

2.11.3 Tor browser

Descargamos desde la página oficial del proyecto Descomprimimos en `~/apps/` y ejecutamos desde terminal:

```
cd ~/apps/tor-browser
./start-tor-browser.desktop --register-app
```

2.11.4 TiddlyDesktop

Descargamos desde la página web, descomprimimos y generamos la entrada en el menú.

2.12 Terminal y Shell

Por defecto tenemos instalado `bash`.

2.12.1 bash-git-promt

Seguimos las instrucciones de este github

2.12.2 zsh

Nos adelantamos a los acontecimientos, pero conviene tener instaladas las herramientas de entornos virtuales de python antes de instalar *zsh* con el plugin para *virtualenvwrapper*.

```
apt install python-all-dev
apt install python3-all-dev
apt install python-pip python-virtualenv virtualenv python3-pip
```

zsh viene por defecto en mi instalación, en caso contrario:

```
apt install zsh
```

Para *zsh* vamos a usar antigen, así que nos lo clonamos en ~/apps/

```
cd ~/apps
git clone https://github.com/zsh-users/antigen
```

También vamos a usar zsh-git-prompt, así que lo clonamos también:

```
cd ~/apps
git clone https://github.com/olivierverdier/zsh-git-prompt)
```

Y editamos el fichero ~/.zshrc para que contenga:

```
# This line loads .profile, it's experimental
[[ -e ~/.profile ]] && emulate sh -c 'source ~/.profile'

source ~/apps/zsh-git-prompt/zshrc.sh
source ~/apps/antigen/antigen.zsh

# Load the oh-my-zsh's library.
antigen use oh-my-zsh

# Bundles from the default repo (robbyrussell's oh-my-zsh).
antigen bundle git
antigen bundle command-not-found

# must install autojump for this
#antigen bundle autojump

# extracts every kind of compressed file
antigen bundle extract

# jump to dir used frequently
antigen bundle z

#antigen bundle pip

antigen bundle common-aliases

antigen bundle robbyrussell/oh-my-zsh plugins/virtualenvwrapper

antigen bundle zsh-users/zsh-completions
```

```
# Syntax highlighting bundle.
antigen bundle zsh-users/zsh-syntax-highlighting
antigen bundle zsh-users/zsh-history-substring-search ./zsh-history-
substring-search.zsh

# Arialdo Martini git needs awesome terminal font
#antigen bundle arialdomartini/oh-my-git
#antigen theme arialdomartini/oh-my-git-themes oppa-lana-style

# autosuggestions
antigen bundle tarruda/zsh-autosuggestions

#antigen theme agnoster
antigen theme gnzh

# Tell antigen that you're done.
antigen apply

# Correct rm alias from common-alias bundle
unalias rm
alias rmi='rm -i'
```

Antigen ya se encarga de descargar todos los plugins que queramos utilizar en zsh. Todos el software se descarga en `~/.antigen`

Para configurar el zsh-git-prompt, que inspiró el bash-git-prompt, he modificado el fichero `~/.zshrc` y el fichero del tema en `~/.antigen/bundles/robbyrussell/oh-my-zsh/themes/gnzh.zsh-theme`

2.12.3 fish

Nota: No he instalado *fish* dejo por aquí las notas del antiguo linux mint por si le interesa a alguien.

Instalamos *fish*:

```
sudo aptitude install fish
```

Instalamos oh-my-fish

```
curl -L https://github.com/oh-my-fish/oh-my-fish/raw/master/bin/install > install
fish install
rm install
```

Si queremos que fish sea nuestro nuevo shell:

```
chsh -s `which fish`
```

Los ficheros de configuración de *fish* se encuentran en `~/config/fish`.

Los ficheros de *Oh-my-fish* en mi portátil quedan en `~/.local/share/omf`

Para tener la info de git en el prompt de fish al estilo de bash-git-prompt, copiamos:

```
cp ~/.bash-git-prompt/gitprompt.fish ~/.config/fish/functions/fish_prompt.fish
```

NOTA: *fish* es un shell estupendo supercómodo con un montón de funcionalidades. Pero no es POSIX. Mucho ojo con esto, usa *fish* pero asegúrate de saber a que renuncias, o las complicaciones a las que vas a enfrentarte.

2.12.4 tmux

Esto no tiene mucho que ver con los shell, lo he instalado para aprender a usarlo.

```
sudo apt install tmux
```

2.13 Utilidades

Agave y *pdftk* ya no existen, nos pasamos a *gpick* y *poppler-utils*:

Instalamos *gpick* con `sudo apt install gpick`

2.14 Codecs

```
sudo apt-get install mint-meta-codecs
```

3 Utilidades

3.1 htop

```
sudo apt install htop
```

3.2 gparted

Instalamos *gparted* para poder formatear memorias usb

```
sudo apt install gparted
```

3.3 wkhtmltopdf

```
sudo apt install wkhtmltopdf
```

4 Internet

5 Rclone

Instalamos desde la página web, siempre que te fies obviamente.

```
curl https://rclone.org/install.sh | sudo bash
```

5.1 Recetas rclone

Copiar directorio local en la nube:

```
rclone copy /localdir hubic:backup -vv
```

Si queremos ver el directorio en la web de Hubic tenemos que copiarlo en *default*:

```
rclone copy /localdir hubic:default/backup -vv
```

Sincronizar una carpeta remota en local:

```
rclone sync hubic:directorio_remoto /home/salvari/directorio_local -vv
```

5.2 Referencias

- Como usar rclone (blogdelazaro)
- y con cifrado (blogdelazaro)
- Documentación

6 Tareas

6.1 hamster-indicator

Tan fácil como:

```
sudo apt install hamster-indicator
```

7 Documentación

7.1 Vanilla LaTeX

El LaTeX de Debian está un poquillo anticuado, si se quiere usar una versión reciente hay que aplicar este truco.

```
cd ~
mkdir tmp
cd tmp
wget http://mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz
tar xzf install-tl-unx.tar.gz
cd install-tl-xxxxxx
```

La parte xxxxxx varía en función del estado de la última versión de LaTeX disponible.

```
sudo ./install-tl
```

Una vez lanzada la instalación podemos desmarcar las opciones que instalan la documentación y las fuentes. Eso nos obligará a consultar la documentación on line pero ahorrará practicamente el 50% del espacio necesario. En mi caso sin doc ni src ocupa 2,3Gb

```
mkdir -p /opt/texbin
sudo ln -s /usr/local/texlive/2018/bin/x86_64-linux/* /opt/texbin
```

Por último para acabar la instalación añadimos `/opt/texbin` al *PATH*. Para *bash* y *zsh* basta con añadir al fichero `~/.profile` las siguientes lineas:

```
# adds texlive to my PATH
if [ -d "/opt/texbin" ] ; then
    PATH="$PATH:/opt/texbin"
fi
```

En cuanto a *fish* (si es que lo usas, claro) tendremos que modificar (o crear) el fichero `~/.config/fish/config.fish` y añadir la siguiente linea:

```
set PATH $PATH /opt/texbin
```

7.1.1 Falsificando paquetes

Ya tenemos el *texlive* instalado, ahora necesitamos que el gestor de paquetes sepa que ya lo tenemos instalado.

```
sudo apt install equivs --no-install-recommends
mkdir -p /tmp/tl-equivs && cd /tmp/tl-equivs
equivs-control texlive-local
```

Alternativamente para hacerlo más fácil podemos descargarnos un fichero `texlive-local`ya preparado, ejecutando:

```
wget http://www.tug.org/texlive/files/debian-equivs-2018-ex.txt
/bin/cp -f debian-equivs-2018-ex.txt texlive-local
```

Editamos la versión (si queremos) y procedemos a generar el paquete *deb*.

```
equivs-build texlive-local
```

El paquete que hemos generado tiene una dependencia: *freeglut3*, hay que instalarla previamente.

```
sudo apt install freeglut3
sudo dpkg -i texlive-local_2018-1_all.deb
```

Todo listo, ahora podemos instalar cualquier paquete debian que dependa de *texlive* sin problemas de dependencias, aunque no hayamos instalado el *texlive* de Debian.

7.1.2 Fuentes

Para dejar disponibles las fuentes opentype y truetype que vienen con texlive para el resto de aplicaciones:

```
sudo cp $(kpsewhich -var-value TEXMFSYSVAR)/fonts/conf/texlive-
fontconfig.conf /etc/fonts/conf.d/09-texlive.conf
gksudo gedit /etc/fonts/conf.d/09-texlive.conf
```

Borramos la linea:

```
<dir>/usr/local/texlive/2018/texmf-dist/fonts/type1</dir>
```

Y ejecutamos:

```
sudo fc-cache -fsv
```

Actualizaciones Para actualizar nuestro *latex* a la última versión de todos los paquetes:

```
sudo /opt/texbin/tlmgr update --self
sudo /opt/texbin/tlmgr update --all
```

También podemos lanzar el instalador gráfico con:

```
sudo /opt/texbin/tlmgr --gui
```

Para usar el instalador gráfico hay que instalar previamente:

```
sudo apt-get install perl-tk --no-install-recommends
```

Lanzador para el actualizador de *texlive*:

```
mkdir -p ~/.local/share/applications
/bin/rm ~/.local/share/applications/tlmgr.desktop
cat > ~/.local/share/applications/tlmgr.desktop << EOF
[Desktop Entry]
Version=1.0
Name=TeX Live Manager
Comment=Manage TeX Live packages
GenericName=Package Manager
Exec=gksu -d -S -D "TeX Live Manager" '/opt/texbin/tlmgr -gui'
Terminal=false
Type=Application
Icon=system-software-update
EOF
```

7.2 Tipos de letra

Creamos el directorio de usuario para tipos de letra:

```
mkdir ~/.local/share/fonts
```

7.3 Fuentes Adicionales

Un par de fuentes las he descargado de internet y las he almacenado en el directorio de usuario para los tipos de letra: `~/.local/share/fonts`

- Ubuntu (La uso en documentación)
- Mensch (Esta es la que yo uso para programar.)

Además he clonado el repo *Programming Fonts* y enlazado algunas fuentes (Hack y Menlo)

```
cd ~/wherever
git clone https://github.com/ProgrammingFonts/ProgrammingFonts
cd ~/.local/share/fonts
ln -s ~/wherever/ProgrammingFonts/Hack .
ln -s ~/wherever/ProgrammingFonts/Menlo .
```

7.4 Pandoc

Pandoc es un traductor entre formatos de documento. Está escrito en Python y es increiblemente útil. De hecho este documento está escrito con *Pandoc*.

Instalado el *Pandoc* descargando paquete deb desde la página web del proyecto.

Además descargamos plantillas adicionales desde este repo ejecutando los siguientes comandos:

```
mkdir ~/.pandoc
cd ~/.pandoc
git clone https://github.com/jgm/pandoc-templates templates
```

7.5 Calibre

La mejor utilidad para gestionar tu colección de libros electrónicos.

Ejecutamos lo que manda la página web:

```
sudo -v && wget -nv -O- https://raw.githubusercontent.com/kovidgoyal/calibre/master/set
installer.py \
| sudo python -c "import sys; main=lambda:sys.stderr.write('Download failed\n'); exec(sy
```

Para usar el calibre con el Kobo Glo:

- Desactivamos todos los plugin de Kobo menos el Kobo Touch Extended
- Creamos una columna MyShelves con identificativo #myshelves
- En las opciones del plugin:
  - En la opción Collection columns añadimos las columnas series,#myshelves

  - Marcamos las opciones Create collections y Delete empy collections
  - Marcamos *Modify CSS*
  - Update metadata on device y Set series information

Algunos enlaces útiles:

- (https://github.com/jgoguen/calibre-kobo-driver)
- (http://www.lectoreselectronicos.com/foro/showthread.php?15116-Manual-de-instalaci%C3%B3n-y-uso-del-plugin-Kobo-Touch-Extended-para-Calibre)
- (http://www.redelijkheid.com/blog/2013/7/25/kobo-glo-ebook-library-management-with-calibre)
- (https://www.netogram.com/kobo.htm)

7.6 Scribus

Scribus es un programa libre de composición de documentos. con Scribus puedes elaborar desde los folletos de una exposición hasta una revista o un poster.

Para tener una versión más actualizada instalamos:

```
sudo add-apt-repository ppa:scribus/ppa
sudo apt update
sudo apt install scribus scribus-ng scribus-template scribus-ng-doc
```

7.6.1 Cambiados algunos valores por defecto

He cambiado los siguientes valores en las dos versiones, non están exactamente en el mismo menú pero no son díficiles de encontrar:

- Lenguaje por defecto: **English**
- Tamaño de documento: **A4**
- Unidades por defecto: **milimeters**
- Show Page Grid: **Activado**
- Dimensiones de la rejilla:
  - Mayor: **30 mm**
  - Menor: **6mm**
- En opciones de salida de *pdf* indicamos que queremos salida a impresora y no a pantalla. Y también que no queremos *spot colors* (que serían solo para ciertas impresoras industriales).

Siempre se puede volver a los valores por defecto sin mucho problema (hay una opción para ello)

Referencia aquí

7.6.2 Solucionados problemas de *hyphenation*

Scribus no hacia correctamente la separación silábica en castellano, he instalado los paquetes:

- hyphen-es
- hyphen-gl

Y ahora funciona correctamente.

8 Desarrollo software

8.1 Paquetes esenciales

Estos son los paquetes esenciales para empezar a desarrollar software en Linux.

```
sudo apt install build-essential checkinstall make automake cmake autoconf git git-core dpkg wget
```

8.2 Git

Control de versiones distribuido. Imprescindible. Para *Linux Mint* viene instalado por defecto.

Configuración básica de git:

```
git config --global ui.color auto
git config --global user.name "Pepito Pérez"
git config --global user.email "pperez@mikasa.com"

git config --global alias.cl clone

git config --global alias.st "status -sb"
git config --global alias.last "log -1 --stat"
git config --global alias.lg "log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %Cblue<%an>%Creset' --abbrev-commit --date=relative --all"
git config --global alias.dc "diff --cached"
```

```
git config --global alias.unstage "reset HEAD --"

git config --global alias.ci commit
git config --global alias.ca "commit -a"

git config --global alias.ri "rebase -i"
git config --global alias.ria "rebase -i --autosquash"
git config --global alias.fix "commit --fixup"
git config --global alias.squ "commit --squash"

git config --global alias.cp cherry-pick
git config --global alias.co checkout
git config --global alias.br branch
git config --global core.editor emacs
```

8.3 Emacs

Instalado emacs desde los repos:

```
sudo aptitude install emacs
```

- Configuramos la fuente por defecto del editor y salvamos las opciones. Con esto generamos el fichero `~/.emacs`
- **Importante**: Configuramos la *face* para la *region* con un color que nos guste. Parece que viene configurado por defecto igual que el texto normal y nunca veremos la *region* resaltada aunque queramos.
- Editamos el fichero `.emacs` y añadimos los depósitos de paquetes (nunca he conseguido que *Marmalade* funcione)

Esta es la sección donde configuramos los depósitos de paquetes:

```
;;-----------------------------------------------------------------
---
;; MELPA and others
(when (>= emacs-major-version 24)
  (require 'package)
  (package-initialize)
 (add-to-list 'package-archives '("melpa" . "http://melpa.org/packages/") t)
 (add-to-list 'package-archives '("gnu" . "http://elpa.gnu.org/packages/") t)
;;  (add-to-list 'package-archives '("marmalade" . "https://marmalade-
repo.org/packages/") t)
  )
```

GNU Elpa es el depósito oficial, tiene menos paquetes y son todos con licencia FSF.

Melpa y Marmalade son paquetes de terceros. Tienen mucha más variedad pero con calidades dispares.

Desde Melpa con el menú de gestión de paquetes de emacs, instalamos los siguientes paquetes:

- *markdown-mode*
- *pandoc-mode*
- *auto-complete*
- *ac-dcd*
- *d-mode*
- *flycheck*
- *flycheck-dmd-dub*
- *flycheck-d-unittest*
- *elpy*
- *jedi*

Después de probar *flymake* y *flycheck* al final me ha gustado más *flycheck* Hay una sección de configuración en el fichero `.emacs` para cada uno de ellos, pero la de *flymake* está comentada.

Configuramos el fichero `.emacs` definimos algunas preferencias, algunas funciones útiles y añadimos orígenes extra de paquetes.

```
(custom-set-variables
 ;; custom-set-variables was added by Custom.
 ;; If you edit it by hand, you could mess it up, so be careful.
 ;; Your init file should contain only one such instance.
 ;; If there is more than one, they won't work right.
 '(show-paren-mode t))
(custom-set-faces
 ;; custom-set-faces was added by Custom.
 ;; If you edit it by hand, you could mess it up, so be careful.
 ;; Your init file should contain only one such instance.
 ;; If there is more than one, they won't work right.
 )

;;-----------------------------------------------------------
;; Some settings
(setq inhibit-startup-message t) ; Eliminate FSF startup msg
```

```
(setq frame-title-format "%b")   ; Put filename in titlebar
;(setq visible-bell t)           ; Flash instead of beep
(set-scroll-bar-mode 'right)     ; Scrollbar placement
(show-paren-mode t)          ; Blinking cursor shows matching parentheses
(setq column-number-mode t)    ; Show column number of current cursor location
(mouse-wheel-mode t)             ; wheel-mouse support

(setq fill-column 78)
(setq auto-fill-mode t)          ; Set line width to 78 columns...

(setq-default indent-tabs-mode nil)     ; Insert spaces instead of tabs
(global-set-key "\r" 'newline-and-indent) ; turn autoindenting on
;(set-default 'truncate-lines t)        ; Truncate lines for all buffers
;(require 'iso-transl)              ; doesn't seems to be needed in debian


;;-------------------------------------------------------------
;; Some useful key definitions
(define-key global-map [M-S-down-mouse-3] 'imenu)
(global-set-key [C-tab] 'hippie-expand)                    ; expand
(global-set-key [C-kp-subtract] 'undo)                     ; [Undo]
(global-set-key [C-kp-multiply] 'goto-line)                ; goto line
(global-set-key [C-kp-add] 'toggle-truncate-lines)         ; goto line
(global-set-key [C-kp-divide] 'delete-trailing-whitespace) ; delete trailing whitespace
(global-set-key [C-kp-decimal] 'completion-at-point)     ; complete at point
(global-set-key [C-M-prior] 'next-buffer)                ; next-buffer
(global-set-key [C-M-next] 'previous-buffer)               ; previous-
buffer

;;-------------------------------------------------------------
;; Set encoding
(prefer-coding-system 'utf-8)
(setq coding-system-for-read 'utf-8)
(setq coding-system-for-write 'utf-8)

;;-------------------------------------------------------------
;; Maximum colors
(cond ((fboundp 'global-font-lock-mode)            ; Turn on font-
lock (syntax highlighting)
     (global-font-lock-mode t)           ; in all modes that support it
```

```
       (setq font-lock-maximum-decoration t))) ; Maximum colors

;;-------------------------------------------------------------
;; Use % to match various kinds of brackets...
;; See: http://www.lifl.fr/~hodique/uploads/Perso/patches.el

(global-set-key "%" 'match-paren)              ; % key match parents
(defun match-paren (arg)
  "Go to the matching paren if on a paren; otherwise insert %."
  (interactive "p")
  (let ((prev-char (char-to-string (preceding-char)))
        (next-char (char-to-string (following-char))))
    (cond ((string-match "[[{(<]" next-char) (forward-sexp 1))
          ((string-match "[\]})>]" prev-char) (backward-sexp 1))
          (t (self-insert-command (or arg 1))))))

;;-------------------------------------------------------------
;; The wonderful bubble-buffer
(defvar LIMIT 1)
(defvar time 0)
(defvar mylist nil)

(defun time-now ()
   (car (cdr (current-time))))

(defun bubble-buffer ()
   (interactive)
   (if (or (> (- (time-now) time) LIMIT) (null mylist))
       (progn (setq mylist (copy-alist (buffer-list)))
          (delq (get-buffer " *Minibuf-0*") mylist)
          (delq (get-buffer " *Minibuf-1*") mylist)))
   (bury-buffer (car mylist))
   (setq mylist (cdr mylist))
   (setq newtop (car mylist))
   (switch-to-buffer (car mylist))
   (setq rest (cdr (copy-alist mylist)))
   (while rest
     (bury-buffer (car rest))
     (setq rest (cdr rest)))
   (setq time (time-now)))
```

```
(global-set-key [f8] 'bubble-buffer)    ; win-tab switch the buffer

(defun geosoft-kill-buffer ()
   ;; Kill default buffer without the extra emacs questions
   (interactive)
   (kill-buffer (buffer-name))
   (set-name))
(global-set-key [C-delete] 'geosoft-kill-buffer)

;;------------------------------------------------------------------
---
;; MELPA and others
(when (>= emacs-major-version 24)
  (require 'package)
  (package-initialize)
 (add-to-list 'package-archives '("melpa" . "http://melpa.org/packages/") t)
 (add-to-list 'package-archives '("gnu" . "http://elpa.gnu.org/packages/") t)
   (add-to-list 'package-archives '("marmalade" . "https://marmalade-
repo.org/packages/") t)
  )

; (add-to-list 'load-path "~/.emacs.d/")

;;------------------------------------------------------------------
---
;; Packages installed via package
;;------------------------------

;;------------------------------------------------------------------
---
;; flymake and flycheck installed from package
;; I think you have to choose only one

;; (require 'flymake)
;;  ;;(global-set-key  (kbd  "C-c  d")  'flymake-display-err-menu-for-
current-line)
;; (global-set-key (kbd "C-c d") 'flymake-popup-current-error-menu)
;; (global-set-key (kbd "C-c n") 'flymake-goto-next-error)
;; (global-set-key (kbd "C-c p") 'flymake-goto-prev-error)
```

```
(add-hook 'after-init-hook #'global-flycheck-mode)
(global-set-key  (kbd "C-c C-p") 'flycheck-previous-error)
(global-set-key  (kbd "C-c C-n") 'flycheck-next-error)

;; Define d-mode addons
;; Activate flymake or flycheck for D
;; Activate auto-complete-mode
;; Activate yasnippet minor mode if available
;; Activate dcd-server
(require 'ac-dcd)
(add-hook 'd-mode-hook
          (lambda()
            ;;(flymake-d-load)
            (flycheck-dmd-dub-set-variables)
            (require 'flycheck-d-unittest)
            (setup-flycheck-d-unittest)
            (auto-complete-mode t)
            (when (featurep 'yasnippet)
              (yas-minor-mode-on))
            (ac-dcd-maybe-start-server)
            (ac-dcd-add-imports)
            (add-to-list 'ac-sources 'ac-source-dcd)
              (define-key d-mode-map (kbd "C-c ?") 'ac-dcd-show-ddoc-
with-buffer)
          (define-key d-mode-map (kbd "C-c .") 'ac-dcd-goto-definition)
            (define-key d-mode-map (kbd "C-c ,") 'ac-dcd-goto-def-pop-
marker)
           (define-key d-mode-map (kbd "C-c s") 'ac-dcd-search-symbol)
            (when (featurep 'popwin)
              (add-to-list 'popwin:special-display-config
                           `(,ac-dcd-error-buffer-name :noselect t))
              (add-to-list 'popwin:special-display-config
                                           `(,ac-dcd-document-buffer-
name :position right :width 80))
              (add-to-list 'popwin:special-display-config
                                         `(,ac-dcd-search-symbol-buffer-
name :position bottom :width 5)))))

;; Define diet template mode (this is not installed from package)
```

```
(add-to-list 'auto-mode-alist '("\\.dt$" . whitespace-mode))
(add-hook 'whitespace-mode-hook
          (lambda()
            (setq tab-width 2)
            (setq whitespace-line-column 250)
            (setq indent-tabs-mode nil)
            (setq indent-line-function 'insert-tab)))

;;-----------------------------------------------------------------
---
;; elpy
(elpy-enable)
```

8.4 Lenguaje de programación D (D programming language)

El lenguaje de programación D es un lenguaje de programación de sistemas con una sintaxis similar a la de C y con tipado estático. Combina eficiencia, control y potencia de modelado con seguridad y productividad.

8.4.1 D-apt e instalación de programas

Configurado *d-apt*, instalados todos los programas incluidos

```
sudo wget http://master.dl.sourceforge.net/project/d-apt/files/d-
apt.list -O /etc/apt/sources.list.d/d-apt.list
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys EBCF975E5BA24D5E
sudo apt update
```

Instalamos todos los programas asociados excepto *textadept* que falla por problemas de librerias.

```
sudo apt install dmd-compiler dmd-tools dub dcd dfix dfmt dscanner
```

8.4.2 DCD

Una vez instalado el DCD tenemos que configurarlo creando el fichero `~/.config/dcd/dcd.conf` con el siguiente contenido:

```
/usr/include/dmd/druntime/import
/usr/include/dmd/phobos
```

Podemos probarlo con:

```
dcd-server &
echo | dcd-client --search toImpl
```

8.4.3 gdc

Instalado con:

```
sudo aptitude install gdc
```

8.4.4 ldc

Instalado con:

```
sudo aptitude install ldc
```

Para poder ejecutar aplicaciones basadas en Vibed, necesitamos instalar:

```
sudo apt-get install -y libssl-dev libevent-dev
```

8.4.5 Emacs para editar D

Instalados los siguientes paquetes desde Melpa

- d-mode
- flymake-d
- flycheck
- flycheck-dmd-dub
- flychek-d-unittest
- auto-complete (desde melpa)
- ac-dcd

Referencias * (https://github.com/atilaneves/ac-dcd) * (https://github.com/Hackerpilot/DCD)

8.5 Processing

Bajamos los paquetes de las respectivas páginas web, descomprimimimos en `~/apps/`, en las nuevas versiones incorpora un script de instalación que ya se encarga de crear el fichero *desktop*.

8.6 Python

De partida tenemos instalado dos versiones: *python* y *python3*

```
python -V
Python 2.7.12

python3 -V
Python 3.5.2
```

8.6.1 Paquetes de desarrollo

Para que no haya problemas a la hora de instalar paquetes en el futuro conviene que instalemos los paquetes de desarrollo:

```
sudo apt install python-dev
sudo apt install python3-dev
```

8.6.2 pip, virtualenv, virtualenvwrapper, virtualfish

Los he instalado a nivel de sistema.

pip es un gestor de paquetes para **Python** que facilita la instalación de librerías y utilidades.

Para poder usar los entornos virtuales instalaremos también *virtualenv*.

Instalamos los dos desde aptitude:

```
sudo apt install python-pip python-virtualenv virtualenv python3-pip
```

virtualenv es una herramienta imprescindible en Python, pero da un poco de trabajo, así que se han desarrollado algunos frontends para simplificar su uso, para *bash* y *zsh* usaremos *virtualenvwrapper*, y para *fish* el *virtualfish*. Como veremos son todos muy parecidos.

Instalamos el virtualwrapper:

```
sudo apt install virtualenvwrapper -y
```

Para usar *virtualenvwrapper* tenemos que hacer:

```
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
```

O añadir esa linea a nuestros ficheros *.bashrc* y/o *.zshrc*

Definimos la variable de entorno *WORKON_HOME* para que apunte al directorio por defecto, `~/.local/share/virtualenvs`. En ese directorio es donde se guardarán nuestros entornos virtuales.

En el fichero `.profile` añadimos:

```
# WORKON_HOME for virtualenvwrapper
if [ -d "$HOME/.local/share/virtualenvs" ] ; then
    WORKON_HOME="$HOME/.local/share/virtualenvs"
fi
```

Aquí tenemos la referencia de comandos de *virtualenvwrapper*

Por último, si queremos tener utilidades parecidas en nuestro *fish shell* instalamos *virtualfish*:

```
sudo pip install virtualfish
```

Aquí tenemos la documentación de *virtualfish* y la descripción de todos los comandos y plugins disponibles.

8.6.3 pipenv

No lo he instalado, pero en caso de instalación mejor lo instalamos a nivel de usuario con:

```
pip install --user pipenv
```

8.6.4 Instalación de bpython y ptpython

bpython instalado desde repos `sudo apt install bpython bpython3`

ptpython instalado en un virtualenv para probarlo

8.6.5 Emacs para programar python

Para instalar `elpy`

```
sudo apt install python-jedi python3-jedi
# flake8 for code checks
sudo apt install flake8 python-flake8 python3-flake8
# and autopep8 for automatic PEP8 formatting
sudo apt install python-autopep8
# and yapf for code formatting
sudo apt install yapf yapf3
```

Añadimos la sección

```
;;-----------------------------------------------------------------
---
;; elpy
```

```
(elpy-enable)
(setq elpy-rpc-backend "jedi")

(add-hook 'python-mode-hook 'jedi:setup)
(setq jedi:complete-on-dot t)
```

Desde *Emacs* ejecutamos: `alt-x jedi:install-server`

8.6.5.1 TODO Estudiar esto con calma https://elpy.readthedocs.io/en/latest

8.6.6 Jupyter

Una instalación para pruebas.

```
mkvirtualenv -p /usr/bin/python3 jupyter
python -m pip install jupyter
```

8.7 neovim

Vamos a probar *neovim*:

```
sudo apt-add-repository ppa:neovim-ppa/stable
sudo apt-get update
sudo apt-get install neovim
```

Para instalar los módulos de python [3]:

```
sudo pip install --upgrade neovim
sudo pip3 install --upgrade neovim
```

Revisar esto

Para actualizar las alternativas:

```
sudo update-alternatives --install /usr/bin/vi vi /usr/bin/nvim 60
sudo update-alternatives --config vi
sudo update-alternatives --install /usr/bin/vim vim /usr/bin/nvim 60
sudo update-alternatives --config vim
```

[3] aun no lo hice

8.7.0.1 Install *vim-plug*

Ejecutamos:

```
curl -fLo ~/.local/share/nvim/site/autoload/plug.vim --create-dirs \
   https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
```

Configuramos el fichero de configuración de *nvim* (`~/.config/nvim/init.vim`):

```
" Specify a directory for plugins
" - For Neovim: ~/.local/share/nvim/plugged
" - Avoid using standard Vim directory names like 'plugin'
call plug#begin('~/.local/share/nvim/plugged')

if has('nvim')
  Plug 'Shougo/deoplete.nvim', { 'do': ':UpdateRemotePlugins' }
else
  Plug 'Shougo/deoplete.nvim'
  Plug 'roxma/nvim-yarp'
  Plug 'roxma/vim-hug-neovim-rpc'
endif
let g:deoplete#enable_at_startup = 1


Plug 'zchee/deoplete-jedi'

" Initialize plugin system
call plug#end()
```

La primera vez que abramos *nvim* tenemos que instalar los plugin por comando ejecutando: `:PlugInstall`

Instalación de `dein`

Solo hay que instalar uno de los dos o *dein* o *plug-vim*. Yo uso *plug-vim* así que esto es sólo una referencia.

https://github.com/Shougo/dein.vim

```
" Add the dein installation directory into runtimepath
set runtimepath+=~/.config/nvim/dein/repos/github.com/Shougo/dein.vim

if dein#load_state('~/.config/nvim/dein')
  call dein#begin('~/.config/nvim/dein')

 call dein#add('~/.config/nvim/dein/repos/github.com/Shougo/dein.vim')
```

```
  call dein#add('Shougo/deoplete.nvim')
  call dein#add('Shougo/denite.nvim')
  if !has('nvim')
    call dein#add('roxma/nvim-yarp')
    call dein#add('roxma/vim-hug-neovim-rpc')
  endif

  call dein#end()
  call dein#save_state()
endif

filetype plugin indent on
syntax enable
```