USB Virus Scanner - RaspberryPI

1 - Introduction

Principe de fonctionnement :

- Une clé USB est insérée dans l'appareil.
- Un menu s'affiche automatiquement.
- Le menu propose soit de formater la clé, soit d'effectuer une analyse anti-virus.
- Il est possible d'afficher la date de la base de donnée des définitions de virus.

Pré-requis matériel :

- Un Nano ordinateur ***Raspberry-pi 3***.
- Une Carte microSD 4Go minimum.
- Une Interface écran/boutons *Pifacecad*. (« Piface Control & Display 2 » chez *Farnell*)

Pré-requis logiciel :

Le système d'exploitation ***Raspbian***.
La librairie ***Pifacecad***
La librairie ***WiringPI***
Le logiciel anti-virus ***ClamAV***

2 - Installation

2.1- L'OS *Raspbian* :

* Télécharger Raspbian https://www.raspberrypi.org/downloads/raspbian/

Il existe actuellement *2016-05-10-raspbian-jessie-lite.zip*, une version qui est entièrement en ligne de commande, sans environnement de bureau, donc très légère, idéale pour une petite carte SD (environ 800Mio une fois installé)

* Déployer l'image sur une carte SD depuis un système *GNU/Linux*, avec la commande DD. (Ou depuis Windows® avec *Win32DiskImager*)

(Attention, *of=/dev/sdx* à adapter pour la cible de votre carte SD, en cas d'erreur il y a risque d'effacement d'un disque dur non désiré)

```
dd bs=4M if=2016-05-10-raspbian-jessie-lite.img of=/dev/sdx; sync
```

* Placer la carte SD dans le *Raspberry*, et le mettre sous tension.

Pour mémoire :

Login : pi

Mot de passe : raspberry [une fois le système en place, le changer !]

* Exécuter *raspi-config* pour étendre le système de fichier à la totalité de la carte SD et configurer les options internationale (locale, timezone, clavier, Wi-Fi)

```
raspi-config
```

* Effectuer les mises à jour :

```
sudo apt-get install rpi-update
sudo rpi-update
sudo apt-get update
sudo apt-get upgrade
```

* Mettre les fichiers temporaires en ram pour économiser sur la durée de vie de la carte SD.

```
sudo nano /etc/fstab
```

```
tmpfs      /tmp         tmpfs   defaults,size=256M  0    0
tmpfs      /var/tmp     tmpfs   defaults,size=256M  0    0
tmpfs      /var/lock    tmpfs   defaults,size=256M  0    0
```

2.2 - Le Wi-Fi :

* Editer le fichier */etc/network/interfaces* et modifier comme suit la section wlan0 :

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet dhcp
      wpa-ssid "le nom du réseau wi-fi"
      wpa-psk "le mot de passe du réseau wi-fi"
```

* Relancer ensuite le réseau avec la commande :

```
service networking restart
```

2.3 - L'Antivirus ClamAV :

* Installation :

```
sudo apt-get install clamav
```

* Pour analyser un disque, avec création de log :

```
sudo clamscan –r --log=/home/pi/autocle/virus.log /home/pi/usb
```

* Mise à jour manuelle des définitions antivirus :

```
sudo freshclam
```

* Mise à jour automatique des définitions de virus :

Le processus de mise à jour *freshclam* a normalement été installé, pour le vérifier, lancer :

```
ps -A | grep clam
```

Le log est stocké dans */var/log/clamav/freshclam.log*

Pour configurer la périodicité de la Mise à jour, éditer : */etc/clamav/freshclam.conf*

Sinon, créer une nouvelle tâche quotidienne avec un script :

```
sudo nano /etc/cron.daily/clamav
```

Le remplir avec ceci :

```
#!/bin/sh
/usr/bin/freshclam >> /var/log/resul_freshclam.txt
```

Rendre le script exécutable :

```
sudo chmod +x /etc/cron.daily/clamav
```

* Pour connaître la date de la base des définitions de virus, voir le log *freshclam.log*

Ou regarder la date du fichier de base */var/lib/clamav/daily.cld*

2.4 - La librairie *Pifacecad* :

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python3-pifacecad
```

Pour fonctionner, *Pifacecad* a besoin qu'on active la gestion des interruptions dans *Raspbian* :

```
sudo raspi-config
```

Choisir l'option N°9 : Advanced Options

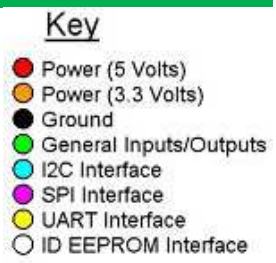
Puis A5 SPI, à modifier pour « Yes »

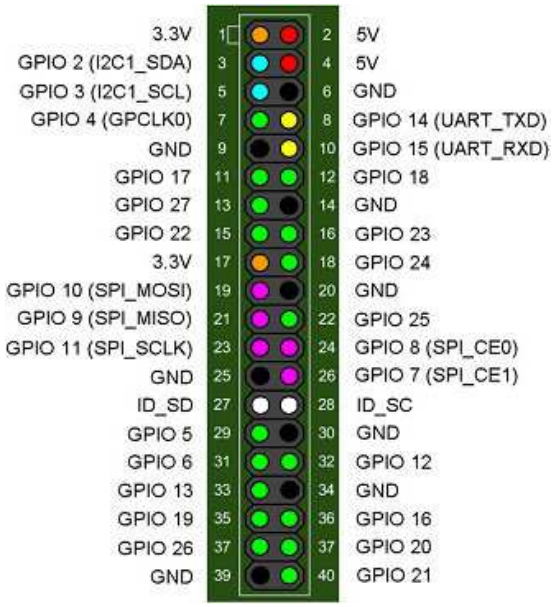
2.5 - La librairie *Wiring PI* :

```
sudo apt-get install git-core
sudo apt-get update
sudo apt-get upgrade
git clone
git://git.drogon.net/wiringPi
cd wiringPi
./build
```

Tester l'installation :

```
gpio -v
gpio readall
```





2.6 - Le Swap File :

Il peut être utile de configurer un gros fichier d'échange pour permettre à clamscan de scanner les gros fichiers.
En effet, ceux qui dépassent environ la taille de la RAM vont causer des erreurs dépassement de mémoire lors du scan anti-virus, trappant ainsi ces fichiers de l'analyse.

Cf. log de clamav : Can't allocate memory ERROR fmap : map allocation failed

Pour passer outre la limitation, on peut configurer la swap de deux manières, soit par le fichier de config, éditer

```
sudo nano /etc/dphys-swapfile
```

Et changer la valeur :

```
CONF_SWAPSIZE=2048
```

Utiliser les commandes suivantes pour prendre en compte le changement :

```
/etc/init.d/dphys-swapfile stop
/etc/init.d/dphys-swapfile start
```

Malheureusement le maximum possible est de 2Go.

Pour passer outre, la seconde méthode consiste à créer une partition swap classique sur la carte SD, puis de la déclarer dans le fstab :

```
sudo nano /etc/fstab
```

Et renseigner cette nouvelle ligne :

```
/dev/mmcblk0p3    swap    swap    defaults    0    0
```

Ainsi les gros fichiers seront pris en compte, et pour autant **clamscan ne se servira pas de ce Swap**.

Une commande existe pour prendre en compte les gros fichiers, mais celle-ci fait swapper le système, ce qui ralentit les performances en plus d'user prématurément la carte SD… Innexploitable !

`clamscan --max-filesize=2000M --max-scansize=2000M` (le maximum est 4Go)

3 - Configurations

3.1 - Pour la détection automatique de l'insertion/éjection de la clé usb :

On utilise des règles UDEV et SystemD custom.

* **Insertion d'une clé :**
Créer le fichier :

```
sudo nano /etc/udev/rules.d/85-usbinsert.rules
```

Y inscrire ceci :

```
ACTION=="add", KERNEL=="sda", ENV{SYSTEMD_WANTS}+="autocle.service"
```

Ainsi, dés l'insertion d'une clé USB (sda) celle-ci sera détectée par le système et le service *autocle.service* sera exécuté. Ce dernier est chargé de lancer le programme *autocle.py*.

Créer le service *autocle.service* :

```
sudo nano /etc/systemd/system/autocle.service
```

Avec dedans :

```
[Unit]
Description=Lance le programme python autocle.py
After=dev-sda.device
BindsTo= dev-sda.device
Requisite= dev-sda.device

[Service]
Type=oneshot
ExecStart=/usr/local/bin/autocle.sh

[Install]
WantedBy=multi-user.target
```

Ne pas installer ce service au démarrage.

* **Éjection d'une clé :**
Créer le fichier :

```
sudo nano /etc/udev/rules.d/20-usbeject.rules
```

Y inscrire ceci :

```
ACTION=="remove", KERNEL=="sda", RUN+="/usr/local/bin/ejectcle.sh"
```

Ainsi, dès l'éjection d'une clé USB (sda) celle-ci sera détectée par le système et le service *autocle.service* sera stoppé.

* Pour recharger les règles UDEV sans redémarrer l'ordi :

```
sudo udevadm control --reload-rules
```

Ou

```
sudo systemctl daemon-reload
```

3.2 - Pour formater la clé USB :

* Installer les pré-requis :

```
sudo apt-get install exfat-utils exfat-fuse ntfs-3g
```

* On utilise la commande suivante :

```
sudo mkfs.exfat /dev/sda1
```

3.3 - Configurer la rotation des fichiers de logs :

Les informations spécifiques à la journalisation sont conservées dans le répertoire *logrotate.d*.
On écrit le fichier suivant :

```
sudo nano /etc/logrotate.d/autocle
```

Avec ceci :

```
/home/pi/autocle/virus.log {
     rotate 12
     weekly
     size 100k
     compress
     delaycompress
     missingok
     create 640  root root
     }
```

Pour s'assurer qu'un fichier de logs effectue correctement ses rotations

```
cat /var/lib/logrotate/status
```

Pour forcer l'exécution :

```
sudo logrotate -f  /etc/logrotate.conf
```

On écrit le fichier suivant :

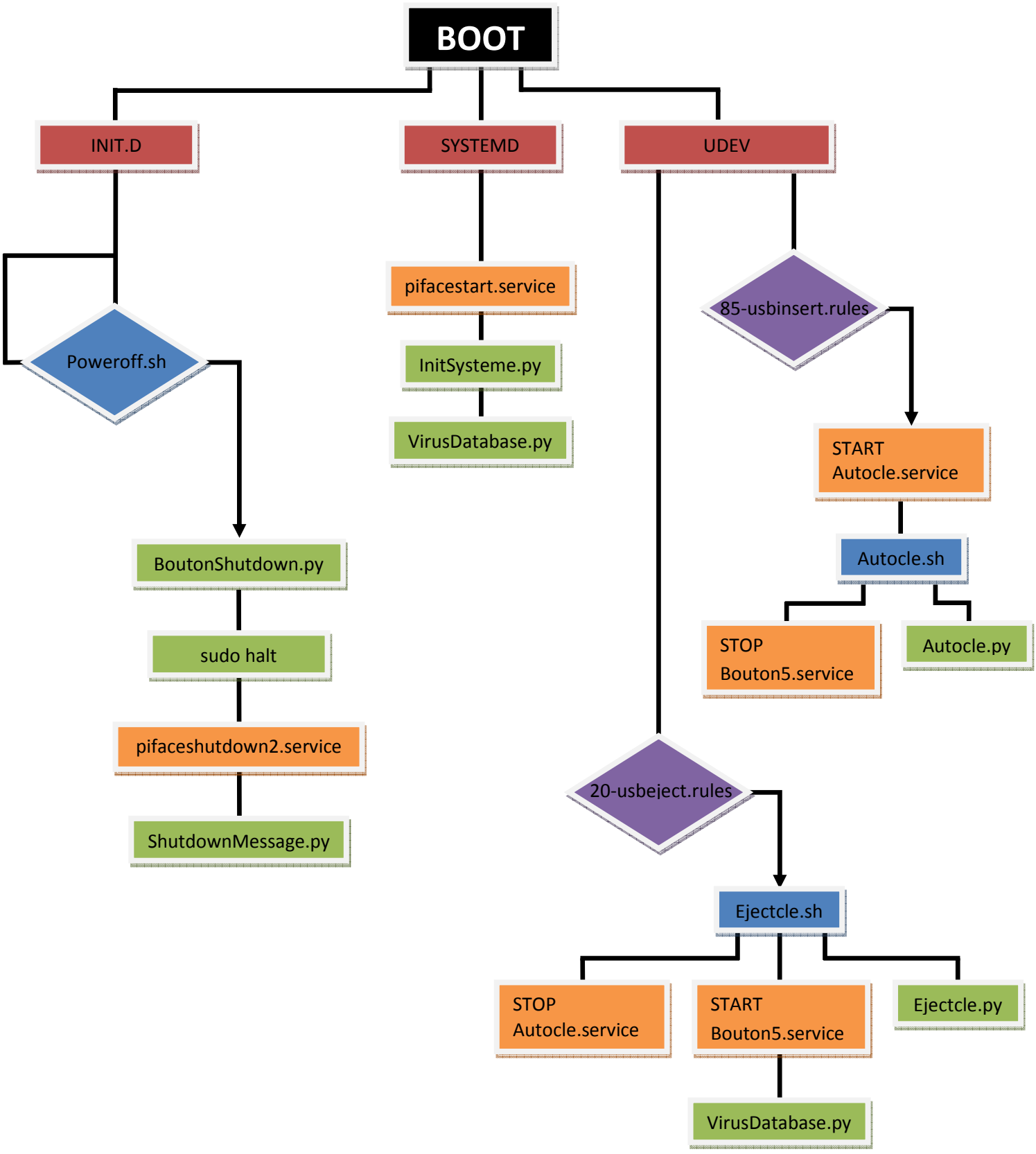
```
sudo nano /etc/logrotate.d/freshclam
```

Avec ceci :

```
/home/pi/autocle/freshclam.log {
     rotate 12
     weekly
     size 100k
     compress
     delaycompress
     missingok
     create 640  root root
     }
```

4 - Les scripts

4.1 - Fonctionnement du système - Scripting :



Chemins complets des scripts :

[Bash]

`/home/pi/autocle/poweroff.sh`	Scrute le *GPI021* et déclenche `BoutonShutdown.py`.
`/usr/local/bin/autocle.sh`	Stoppe le *Bouton5.service* et lance `autocle.py`.
`/usr/local/bin/ejectcle.sh`	Stoppe *autocle.service,* lance `ejectcle.py` et *Bouton5.service*.

[Python]

`/home/pi/autocle/autocle.py`	Programme principal.
`/home/pi/autocle/BoutonShutdown.py`	Affiche « *le système va s'éteindre* » et halt.
`/home/pi/autocle/ejectcle.py`	Efface et éteint l'écran.
`/home/pi/autocle/InitSysteme.py`	Affiche « *démarrage en cours* » et lance `VirusDatabase.py`.
`/home/pi/autocle/ShutdownMessage.py`	Affiche « *attendre 30 Sec pour debrancher* ».
`/home/pi/autocle/VirusDatabase.py`	Affiche la date des déf de virus puis « *système opérationnel* ».

4.2 - Mise en place du Shutdown Message :

* Écrire le script python suivant :

```
nano/home/pi/autocle/ShutdownMessage.py
```

Avec dedans :

```
import pifacecad as p

# pour afficher le message texte

cad = p.PiFaceCAD()
cad.lcd.backlight_on()
cad.lcd.cursor_off()
cad.lcd.blink_off()
cad.lcd.write("Attendre 30 Sec\npour debrancher")
```

* Pour que ce message s'affiche automatiquement quelques secondes avant le halt de l'ordinateur :

Créer le service `pifaceshutdown.service` :

```
sudo nano /etc/systemd/system/pifaceshutdown2.service
```

Avec dedans :

```
[Unit]
Description=Affiche un message sur le piface à l'arrêt du rasberry
Before=shutdown.target

[Service]
ExecStart=/bin/true
ExecStop=/usr/bin/python3 /home/pi/autocle/ShutdownMessage.py
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

Installer le service au démarrage :

```
sudo systemctl --system daemon-reload
sudo systemctl enable pifaceshutdown2.service
```

Pour manipuler le service :

```
sudo systemctl status pifaceshutdown2.service
sudo systemctl start pifaceshutdown2.service
```

4.3 - Script `InitSysteme.py` :

Écrire le script python suivant :

```
nano /home/pi/autocle/InitSysteme.py
```

Avec :

```
import pifacecad  # module de l'afficheur
import time # module pour les fonctions liée à la gestion du temps
import os   # module du systéme linux

#time.sleep(4)    # pour ajuster l'affichage pile sur le démarrage


cad = pifacecad.PiFaceCAD()   # init fonction systéme libraire pifacecad

cad.lcd.clear()         # efface l'écran
cad.lcd.backlight_on()  # active le rétroéclairage de l'afficheur
cad.lcd.write("Demarrage en\ncours...")  # affiche le texte
time.sleep(15)    # pour attendre 10 secondes
cad.lcd.clear()   # efface l'écran
cad.lcd.backlight_off() # active le rétroéclairage de l'afficheur
cad.lcd.cursor_off()    # désactive le curseur texte
cad.lcd.blink_off()     # désactive le carré clignottant
os.system("python3 /home/pi/autocle/VirusDatabase.py")
```

* Pour que ce message s'affiche automatiquement au démarrage de l'ordinateur :

Créer le service `pifacestart.service` :

```
sudo nano /etc/systemd/system/pifacestart.service
```

Avec dedans :

```
[Unit]
Description=Affiche un message sur le piface au démarrage du rasberry

[Service]
ExecStart=/usr/bin/python3 /home/pi/autocle/InitSysteme.py

[Install]
WantedBy=multi-user.target
```

Installer le service au démarrage :

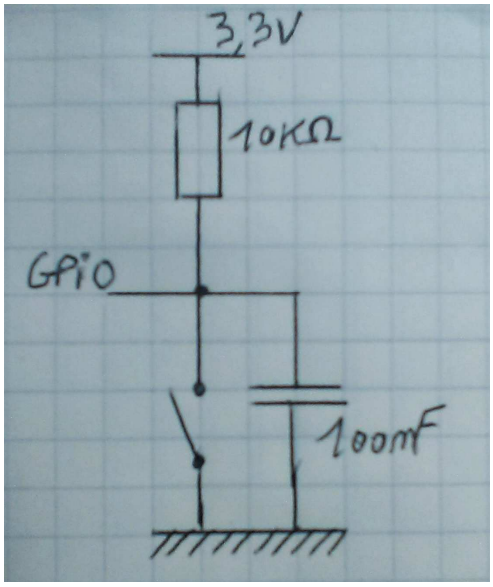
```
sudo systemctl --system daemon-reload
sudo systemctl enable pifacestart.service
```

Pour manipuler le service :

```
sudo systemctl status pifacestart.service
sudo systemctl start pifacestart.service
```

4.4 - Bouton power OFF :

Câbler un bouton poussoir sur le port GPIO21



Écrire le script bash suivant :

```
nano /home/pi/autocle/poweroff.sh
```

Avec dedans :

```
#!/bin/bash

echo "21" > /sys/class/gpio/export  # Pour créer le port gpio logiciel
sleep 3s    # temporisation pour permettre au port logiciel d'avoir le temps
de se créer via la ligne précédente
echo "in" > /sys/class/gpio/gpio21/direction   # pour définir en entrée le
gpio
echo "high" > /sys/class/gpio/gpio21/direction # pour forcer un état haut sur
le gpio

while [ true ]    # Tant que c'est vrai (boucle infinie)
do    # faire
if [ "$(cat /sys/class/gpio/gpio21/value)" == '0' ]  # lire la valeur du gpio,
si = 0
then  # alors
            python3 /home/pi/autocle/BoutonShutdown.py     # exécuter le
script python3
            exit 0
fi
sleep 1
done
```

Rendre le script exécutable :

```
sudo chmod+x /home/pi/autocle/poweroff.sh
```

Pour exécuter le script au démarrage, éditer le fichier :

```
sudo nano /etc/rc.local
```

Et inscrire le chemin vers le script avec un & à la fin, avant la ligne `exit 0`.

```
/home/pi/autocle/poweroff.sh &
```

4.5 - Script BoutonShutdown.py :

Écrire le script python suivant :

```
nano /home/pi/autocle/BoutonShutdown.py
```

Avec :

```
import pifacecad # module de l'afficheur
import os # module du systéme linux
import time

cad = pifacecad.PiFaceCAD() # init fonction système libraire pifacecad

cad.lcd.clear()   # efface l'écran
cad.lcd.backlight_on()  # active le rétroéclairage de l'afficheur
cad.lcd.cursor_off()    # désactive le curseur texte
cad.lcd.blink_on()      # active le carré clignottant
cad.lcd.write("Le systeme va\ns'eteindre")     # affiche le texte
time.sleep(3)
os.system("sudo halt")  # appelle la commande linux halt pour arrêter le
système
```

4.6 - Script ejectcle.sh et ejectcle.py :

Écrire le script bash suivant :

```
sudo nano /usr/local/bin/ejectcle.sh
```

Avec :

```
#!/bin/bash

python3 /home/pi/autocle/ejectcle.py
systemctl stop autocle.service
systemctl start Bouton5.service
```

Rendre le script exécutable :

```
sudo chmod +x /usr/local/bin/ejectcle.sh
```

Écrire le script python suivant :

```
nano /home/pi/autocle/ejectcle.py
```

Avec :

```
import pifacecad

cad = pifacecad.PiFaceCAD()

cad.lcd.backlight_off()
cad.lcd.cursor_off()
cad.lcd.blink_off()
```

4.7 - Script `autocle.sh et autocle.py` :

Écrire le script bash suivant :

```
sudo nano /usr/local/bin/autocle.sh
```

Avec :

```
#!/bin/bash

sudo systemctl stop Bouton5.service
sudo python3 /home/pi/autocle/autocle.py
```

Rendre le script exécutable :

```
sudo chmod +x /usr/local/bin/autocle.sh
```

Créer le dossier pour le point de montage de la cléUSB */home/pi/usb*

```
mkdir /home/pi/usb
```

Écrire le script python suivant :

```
nano /home/pi/autocle/autocle.py
```

Avec :

```
# coding: utf-8

# import des module nécessaire
import pifacecad  # module de l'afficheur
import os, time   # module du systéme linux

# Variables
cad = pifacecad.PiFaceCAD()   # init fonction systéme libraire pifacecad


# Déclaration des Fonctions
def checkcle():
    cad.lcd.backlight_on()
    cad.lcd.blink_off() # désactive le carré clignottant
    cad.lcd.clear()     # efface l'écran
    cad.lcd.cursor_off()      # désactive le curseur texte
    os.system("sudo fdisk -l | grep sda1 > checkcle")
    checkcle= open('checkcle',"r")
    for lignes in checkcle:
        if "NTFS/exFAT" in lignes:
            line = lignes.split()
            cad.lcd.write("Cle en NTFSexFAT")  # affiche le texte
            time.sleep(1)
 #           print("résultat:",line[6])
            cad.lcd.clear()
            cad.lcd.write("Votre choix :\n>Formater  >Scan")     # affiche le
texte sur l'afficheur
            listener.register(0, pifacecad.IODIR_ON, formater)   # appel de la
fonction register, paramétre 0=N° du switch, paramétre IODIR_ON sur l'appuie
du bouton, appelle la focntion formater)
            listener.register(4, pifacecad.IODIR_ON, scanner)
        elif "FAT16" in lignes:
            line = lignes.split()
            cad.lcd.write("Cle en FAT16")# affiche le texte
            time.sleep(1)
#           print("résultat:",line[6])
            cad.lcd.clear()
            cad.lcd.write("Votre choix :\n>Formater  >Scan")     # affiche le
texte sur l'afficheur
            listener.register(0, pifacecad.IODIR_ON, formater)   # appel de la
fonction register, paramétre 0=N° du switch, paramétre IODIR_ON sur l'appuie
```

```
du bouton, appelle la focntion formater)
            listener.register(4, pifacecad.IODIR_ON, scanner)
        elif "FAT32" in lignes:
            line = lignes.split()
            cad.lcd.write("Cle en FAT32")# affiche le texte
            time.sleep(1)
 #           print("résultat:",line[7])
            cad.lcd.clear()
            cad.lcd.write("Votre choix :\n>Formater  >Scan")     # affiche le
texte sur l'afficheur
            listener.register(0, pifacecad.IODIR_ON, formater)   # appel de la
fonction register, paramétre 0=N° du switch, paramétre IODIR_ON sur l'appuie
du bouton, appelle la focntion formater)
            listener.register(4, pifacecad.IODIR_ON, scanner)
        else:
            cad.lcd.write("Cle imcompatible\n>Formater ?") # affiche le texte
            listener.register(0, pifacecad.IODIR_ON, formater)   # appel de la
fonction register, paramétre 0=N° du switch, paramétre IODIR_ON sur l'appuie
du bouton, appelle la focntion formater)


def partition():  # La cle est partitionnée ou non ?
#    listener.register(5, pifacecad.IODIR_ON, rien)  # appel de la fonction
register, paramétre 0=N° du switch, paramétre IODIR_ON sur l'appuie du bouton,
appelle la focntion formater)

    identcle=os.popen("ls /dev/sda1").read().splitlines()  # teste si la cle
est partionnee ou non
    if identcle == ['/dev/sda1']:  # si la cle est partionnee
        checkcle()      # appelle la fonction checkcle
    else:   # sinon proposer le formatage
        cad.lcd.backlight_on()      # allume le rétro-éclairage de l'écran
        cad.lcd.clear()       # efface l'écran
        cad.lcd.blink_off()   # désactive le carré clignottant
        cad.lcd.cursor_off()  # désactive le curseur texte
        cad.lcd.write("Cle incompatible !! :\n>Formater ?")      # affiche le
texte sur l'afficheur
        listener.register(0, pifacecad.IODIR_ON, formater) # appel de la
fonction register, paramétre 0=N° du switch, paramétre IODIR_ON sur l'appuie
du bouton, appelle la focntion formater)
  #      time.sleep(5)

#def oui(event):
#    variable = 1
#    print("variable",variable)
#def non(event):
#    variable = 2
#    print("variable",variable)


def formater(event):    # définition  de la fonction formater
    variable = 0
    cad.lcd.clear()     # efface l'écran
    cad.lcd.write("etes-vous sur ?\n >Oui >Non")     # affiche le texte
    while cad.switch_port.value not in (2,8):
        print("variable :",variable)
    variable = cad.switch_port.value
    if variable == 2:
        formateroui()
    elif variable == 8:
        formaternon()


def formateroui():
    cad.lcd.clear()     # efface l'écran
    cad.lcd.cursor_off()      # désactive le curseur texte
    cad.lcd.blink_on()  # active le carré clignottant
    cad.lcd.write("Formatage\nen cours...")    # affiche le texte
```

```
 #   os.system("date=$(date +%d-%m-%Y)")
    os.system("echo ',,7;' | sudo sfdisk /dev/sda")  # creer une partition
unique de type c Win95 FAT32 (LBA)
    os.system("var=$(date +%d-%m-%Y) ; mkfs.exfat /dev/sda1 -n Cle$var")   #
appelle cette commande linux de formatage
    cad.lcd.clear()     # efface l'écran
    cad.lcd.cursor_off()      # désactive le curseur texte
    cad.lcd.blink_off() # désactive le carré clignottant
    cad.lcd.write("Formatage ok\nRetirer la cle->")  # affiche le texte
    os.system("pkill -f autocle.py")     # appelle la commande linux pour tuer
tous les processus python3, afin de libérer la main et permettre à UDEV d'être
fonctionnel tout de suite sans attendre la minute de délais de libération
standard
#    os.system("killall -9 python3")     # appelle la commande linux pour tuer
tous les processus python3, afin de libérer la main et permettre à UDEV d'être
fonctionnel tout de suite sans attendre la minute de délais de libération
standard

def formaternon():
    cad.lcd.clear()     # efface l'écran
    cad.lcd.cursor_off()      # désactive le curseur texte
    cad.lcd.blink_off() # désactive le carré clignottant
    cad.lcd.write("Annulation\nRetirer la cle->")    # affiche le texte
    os.system("pkill -f autocle.py")     # appelle la commande linux pour tuer
tous les processus python3, afin de libérer la main et permettre à UDEV d'être
fonctionnel tout de suite sans attendre la minute de délais de libération
standard
#    os.system("killall -9 python3")     # appelle la commande linux pour tuer
tous les processus python3, afin de libérer la main et permettre à UDEV d'être
fonctionnel tout de suite sans attendre la minute de délais de libération
standard

def scanner(event): # définition de la fonction scanner
    cad.lcd.clear()
    cad.lcd.cursor_off()
    cad.lcd.blink_on()
    cad.lcd.write("Scan anti-virus\nen cours...")
    os.system("mount /dev/sda1 /home/pi/usb")  # monter la la partition de la
clé USB
    os.system("clamscan -r --log=/home/pi/autocle/virus.log /home/pi/usb")   #
appelle l'anti-virus clamav
    os.system("umount /dev/sda1")  # pour démonter partition de la clé USB
    cad.lcd.clear()
    cad.lcd.blink_off()
    cad.lcd.cursor_off()
    rapport()     # appelle la focntion rapport
    cad.lcd.write("Scan ok\nRetirer la cle->")
    os.system("pkill -f autocle.py")
#    os.system("killall -9 python3")     # appelle la commande linux pour tuer
tous les processus python3, afin de libérer la main et permettre à UDEV d'être
fonctionnel tout de suite sans attendre la minute de délais de libération
standard

def rapport():    # définition de la fonction pour afficher le rapport de
virus

# Ouvrir le fichier de log (de clamav) et inveser les lignes de son contenu
(la fin, au début)
    fichierlog = open("/home/pi/autocle/virus.log", "r")   # ouvre le fichier
en lecture
    contenulog = fichierlog.readlines()  # appelle la fonction readlines pour
lire toutes les lignes du fichier et les mettre dans la variable contenu
    fichierlog.close()  # ferme le fichier
    contenulog.reverse()     # appelle fonction reverse appliqué à la variable
contenu

    tempolog = open("/home/pi/autocle/tmp.txt", "w") # ouvre/crée le fichier
en écriture
```

```
    for line in contenulog:   # pour chaque ligne de la variable contenu
        tempolog.write(line)  # écrire les lignes dans tempolog, càd le fichier
tmp.txt
    tempolog.close()    # fermer le fichier txt

    tempolog = open("/home/pi/autocle/tmp.txt", "r") # ouvre le fichier en
lecture
    ligne=tempolog.readline()# lit la première ligne du fichier tempolog et
stocke dans la variable ligne
    ligne=tempolog.readline()   # lit la deuxième ligne
    ligne=tempolog.readline()   # lit la troisième ligne
    ligne=tempolog.readline()   # lit la quatrième ligne
    print(ligne)    # affiche la dernière ligne lue dans tty1
    cad.lcd.write(ligne.replace(" ","")) # pour supprimer tous les espaces
dans le texte et l'afficher sur l'écran
    tempolog.close()    # fermer le fichier txt


# programme
os.system("pkill -f VirusDatabase.py")
listener = pifacecad.SwitchEventListener()     # init du mode interruption des
switchs
cad.lcd.backlight_on()  # active le rétroéclairage de l'afficheur
partition()
listener.activate()     # active la fonction d'écoute des interruptions
```

4.8 - Script `VirusDatabase.py` :

Créer le service *Bouton5.service* :

```
sudo nano /etc/systemd/system/Bouton5.service
```

Avec dedans :

```
[Unit]
Description=Lance et relance la scrutation du bouton5 pour lancer le script
python associé

[Service]
ExecStart=/usr/bin/python3 /home/pi/autocle/VirusDatabase.py

[Install]
WantedBy=multi-user.target
```

Ne pas installer ce service au démarrage.

Écrire le script python suivant :

```
nano /home/pi/autocle/VirusDatabase.py
```

Avec :

```
#!/usr/bin/env python3
#_*_coding:utf-8_*_

import pifacecad  # module de l'afficheur
import time # module pour les fonctions liée à la gestion du temps
import os   # module du systéme linux
import os.path

cad = pifacecad.PiFaceCAD()   # init fonction systéme libraire pifacecad

def bouton5():
    listener.register(5, pifacecad.IODIR_ON, VirusDatabase)

def VirusDatabase(event):
    if os.path.isfile('/var/lib/clamav/daily.cvd'):
        database = time.ctime(os.path.getmtime('/var/lib/clamav/daily.cvd')) #
récupére la date de modification du fichier de définitions de virus et la
stocke dans la variable
    else:
        database = time.ctime(os.path.getmtime('/var/lib/clamav/daily.cld')) #
récupére la date de modification du fichier de définitions de virus et la
stocke dans la variable
    #print (database)
    cad.lcd.clear()     # efface l'écran
    cad.lcd.backlight_on()    # active le rétroéclairage de l'afficheur
    cad.lcd.cursor_off()      # désactive le curseur texte
    cad.lcd.blink_off() # désactive le carré clignottant
    cad.lcd.write("Virus
Database\n"+database[8:10]+database[4:7]+database[20:24]+database[10:16])
    time.sleep(6) # pour attendre 6 secondes
    cad.lcd.clear()     # efface l'écran
    cad.lcd.write("Systeme\nOperationnel")     # affiche le texte
    cad.lcd.backlight_off()   # active le rétroéclairage de l'afficheur
    cad.lcd.cursor_off()      # désactive le curseur texte
    cad.lcd.blink_off() # désactive le carré clignottant

# main
listener = pifacecad.SwitchEventListener()
cad.lcd.cursor_off()    # désactive le curseur texte
cad.lcd.blink_off()     # désactive le carré clignottant
cad.lcd.write("Systeme\nOperationnel")   # affiche le texte
bouton5()
listener.activate()
```

5 - Simulation

5.1- Pour tester le concept dans la console *GNU/Linux* :

* On peut utiliser le script suivant :

```
#!/bin/bash


echo -n "Que voulez-vous faire ?"
echo ""
echo "- 1 - Formater la clé"
echo "- 2 - Analyser la clé"

echo -n "Votre choix :"

read choix

case "$choix" in
            1) sudo mkfs.vfat /dev/sda1
            echo ""
            echo "la clé est maintenant vierge"
            ;;

            2) sudo mount /dev/sda1 /home/pi/usb
            sudo clamscan -r --log=/var/log/clamav/virus.log /home/pi/usb
            sudo umount /dev/sda1
            echo "la clé est maintenant saine"
            ;;
esac
```

Ressources :

http://piface.github.io/pifacecad/
http://www.piface.org.uk/guides/
http://www.piface.org.uk/guides/setting_up_pifacecad/running_pifacecad_sysinfo/
http://piface.github.io/pifacecad/included_examples.html
http://raspi.tv/2013/how-to-use-interrupts-with-python-on-the-raspberry-pi-and-rpi-gpio-part-2
http://swindon.hackspace.org.uk/blog/piface-shutdown-message/
http://wiringpi.com/download-and-install/
https://embeddedcode.wordpress.com/2013/10/18/adding-a-shutdown-button-to-the-raspberry-pi/
http://www.developpez.net/forums/d563544/systemes/linux/administration-systeme/connaitre-type-d-partition/
https://forum.ubuntu-fr.org/viewtopic.php?id=430231
https://suntong.github.io/blogs/2015/12/25/use-sfdisk-to-partition-disks/
https://wiki.archlinux.org/index.php/systemd#Writing_unit_files
http://blog.tjll.net/systemd-for-device-activation-and-media-archiving/
http://blog.fraggod.net/2012/06/16/proper-ish-way-to-start-long-running-systemd-service-on-udev-event-device-hotplug.html
http://jasonwryan.com/blog/2014/01/20/udev/
http://ktaraghi.blogspot.fr/2013/11/what-is-systemd-and-how-it-works-part-1.html
http://ktaraghi.blogspot.fr/2013/12/what-is-systemd-and-how-it-works-part-2.html
http://ktaraghi.blogspot.fr/2014/01/what-is-systemd-and-how-it-works-part-3.html
https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files
https://bbs.archlinux.org/viewtopic.php?id=181080
https://bbs.archlinux.org/viewtopic.php?id=149419
http://www.framboise314.fr/systemd-tout-nouveau-tout-beau-ou-pas/
https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/sect-Managing_Services_with_systemd-Unit_Files.html