Activiy-watchdog

Script en BASH per controlar l'execució de processos en un sistema Linux.

Creat per: Nahum Manuel Martín Vegas

Versió: 1.0

Data: 04-dec-2019

URL: https://github.com/nadamas2000/activity-watchdog

Aquest script s'ha pensat per controlar l'execució de processos dins d'un sistema Linux i actuar si algun dels processos supervisats no compleix amb els requeriments d'execució.

Es va conceptualitzar per fer-ho servir des de l'inici del sistema dintre de la configuració de Cron.

Per a que arrenqui amb el sistema s'ha d'entrar a la configuració de Cron amb:

- sudo crontab –e

I editar la configuració afegint la línia:

@reboot /"dir"/activity-watchdog.sh

També es pot fer servir iniciant-ho de forma puntual executant directament el fitxer activity-watchdog.sh

El programa consta de quatre parts, cinc fitxers i dos fitxers més generats de forma automàtica.

Les parts del programa son:

- Inici, configuració i execució perpetua del programa que consta de dos fitxers: activity-watchdog.sh i activity-watchdog.conf. Com pot deduir, la configuració de tot el programa es troba a l'arxiu activity-watchdog.conf i l'executable permanent es activity-watchdog.sh.
- Control de processos "forçats": Es troba en part al fitxer software-watchdog.shlib. Aquest fitxer conté funcions per verificar aquells processos que es necessiten timbre actius.
- Control de processos en llista negra: Es troba en part al fitxer software-watchdog.shlib. Aquest fitxer conté funcions per verificar els processos de una llista negra que no s'han d'estar executant. Així com la llista d'excepcions a la llista negra.
- Activitat d'usuari: Es troba al fitxer idle-watchdog.shlib. Aquesta funcionalitat s'encarrega de verificar l'activitat de l'usuari a l'entorn X i quan passa cert tems designat d'inactivitat es finalitzen els processos especificats.

Els fitxers .shlib no son executables, sinó que contenen funcions que son importades al programa principal activity-watchdog.sh

Els fitxers

Com es comenta abans, els fitxers principals del programa son:

- activity-watchdog.sh
- activity-watchdog.conf
- software-watchdog.conf
- idle-watchdog.conf

Els fitxers addicionals que pot generar el programa son:

- activity-watchdog.log
- idle-watchdog.time

El primer fitxer addicional “activity-watchdog.log” conté un registre dels esdeveniments que van succeint en el programa i per defecte es configura per sobreescriure el fitxer en cada execució del programa. O pot emmagatzemar la informació de totes les execucions sense esborrar les dades anteriors si es canvia la configuració.

```
Mon 9 Dec 08:11:15 CET 2019 Creating new logfile...
Mon 9 Dec 08:11:15 CET 2019 Waiting for the first execution of Xorg ...
Mon 9 Dec 08:11:25 CET 2019 Xorg detected with PID: 1647
Mon 9 Dec 08:11:25 CET 2019 Waiting 20
Mon 9 Dec 08:12:06 CET 2019 Initializing watchdog (while-true)
Mon 9 Dec 08:12:14 CET 2019 chromium-browse is not running
Mon 9 Dec 08:12:14 CET 2019 Executing /home/soft/bin/start-chrome-pi.sh
Mon 9 Dec 08:12:36 CET 2019 chromium-browse restarted
Mon 9 Dec 08:14:04 CET 2019 Idle time exceded: 61003
Mon 9 Dec 08:14:04 CET 2019 Command to kill chromium-browse executed
Mon 9 Dec 08:14:10 CET 2019 chromium-browse is not running
Mon 9 Dec 08:14:10 CET 2019 Executing /home/soft/bin/start-chrome-pi.sh
Mon 9 Dec 08:14:13 CET 2019 chromium-browse restarted
```

El segon fitxer addicional conté el registre de temps d’activitat de l’usuari mostrant la data, el temps d’inactivitat, els temps límits i el condicionament actual d’actuació. Aquest arxiu sempre es sobreescriu

```
Mon  9 Dec 09:10:25 CET 2019
Idle time: 3442059 = 0h 57m 22s
Idle max: 1 m
Idle stop: 1 m 5 s
Watchdog down (idle > idle_stop): 3442059 > 65000
```

Configuració

El fitxer de configuració consta de quatre parts: General configuration, Force-list software configuration, Blacklist watchdog configuration i Idle watchdog configuration.

La primera fa referencia a la part general del programa i consta dels següents paràmetres:

- USER: Que conté l'usuari al qual hem de supervisar l'execució dels programes. Si es volen supervisar diferents usuaris s'han de instanciar diferents execucions del programa en diferents carpetes amb el seu propi fitxer de configuració. Per defecte el seu valor es "username". Com exemple si l'usuari a controlar es diu "pi" posarem el paràmetre com:
  - **USER="pi"**
- LOGFILEDIR: És el directori on es pot guardar el fitxer de registre "activity-watchdog.log". Per defecte el paràmetre es el mateix que $EXEDIR que es el directori on es troba el propi executable "activity-watchdog.sh". Es pot canviar per un directori personalitzat con "/var/logs" (sense la ultima barra "/").
  - **LOGFILEDIR=$EXEDIR**
- LOGFILENAME: És el nom del fitxer de registre d'activitat de programa. Es pot personalitzar o inclús es pot deshabilitar redirigint-ho cap a /dev/null on el LOGFILEDIR haurà de establir-se com "/dev" i el nom del fitxer com "null". El paràmetre per defecte es:
  - **LOGFILENAME="activity-watchdog.log"**
- LOGFILE_BY_EXECUTION: Permet activar la funció de sobreescriptura del fitxer de registre. Això esborrarà tota la informació anterior a la no va execució del programa al fitxer de registre. Si es desactiva la informació s'acumularà indefinidament. Per defecte el seu valor es actiu:
  - **LOGFILE_BY_EXECUTION=true**
- WAITING_FOR_PROC: Es el procés que activa el programa, fins que el procés especificat no sigui actiu el programa de supervisió no s'activarà. El valor per defecte es :
  - **WAITING_FOR_PROC="Xorg"**
- FIRST_WAITING: És el temps d'espera després de detectar l'aparició del procés especificat a WAITING_FOR_PROC. Conceptualment es un temps d'espera de càrrega del procés i post processos fins que la situació sigui idònia per la supervisió. Per defecte son 20 segons:
  - **FIRST_WAITING=20**
- OBS_INTERVAL: És l'interval de temps entre un cicle de revisió de processos i un altre. El temps mínim i per defecte es de 1 segon:
  - **OBS_INTERVAL=1**
- IDLE_WATCHDOG_ENABLED: Activa o desactiva la funció idle-watchdog. Si no es farà servir aquesta funcionalitat es millor desactivar-la. Activat per defecte.
  - **IDLE_WATCHDOG_ENABLED=true**
- FORCE_LIST_WATCHDOG_ENABLED: Activa o desactiva la funció force- list-watchdog. Si no es farà servir aquesta funcionalitat es millor desactivar-la. Activat per defecte.
  - **FORCE_LIST_WATCHDOG_ENABLED=true**
- BLACKLIST_WATCHDOG_ENABLED: Activa o desactiva la funció blacklist-watchdog. Si no es farà servir aquesta funcionalitat es millor desactivar-la. Activat per defecte.
  - **BLACKLIST_WATCHDOG_ENABLED=true**

La segona part es refereix a la configuració de Force List Watchdog:

- FORCE_LIST_PROC: És la llista de processos que per defecte han de ser actius o executats de forma persistent. La es composa de parelles con la primera component es refereix al nom del procés i la segona a la instrucció o script que s'ha d'executar si el procés no hi es actiu. Opcionalment es poden separar els processos en línies amb una barra "\" sense cometes i separada amb espai després de cada parella "procés"-"execució". Per defecte la llista es buida. Un exemple seria:
    - **FORCE_LIST_PROC: ("chromium-browse" "/home/soft/bin/start-chrome-pi.sh")**

La tercera part fa referencia a la Blacklist Watchdog:

- GENERAL_VIOLATION_PROCEDURE: Es un procediment general que es pot definir per executar-se si s'executa un procediment llistat a la Blacklist. Per defecte es buit "". Un exemple és el següent:
    - **GENERAL_VIOLATION_PROCEDURE="/home/soft/bin/reiniciarX.sh"**
- BLACKLIST_PROC: Conté la llista de processos que no han de executar-se en parelles amb el procediment d'execució. Per defecte el seu valor d'execució per violació ha de ser $GENERAL_VIOLATION_PROCEDURE. La llista es buida per defecte. Un exemple pot ser:
    - **BLACKLIST_PROC=("pcmanfm" $GENERAL_VIOLATION_PROCEDURE **
      **"vi" $GENERAL_VIOLATION_PROCEDURE **
      **"vim" "/home/soft/bin/reiniciarX.sh" **
      **"nano" $GENERAL_VIOLATION_PROCEDURE **
      **"lxterminal" $GENERAL_VIOLATION_PROCEDURE **
      **"xterm" $GENERAL_VIOLATION_PROCEDURE **
      "uxterm" $GENERAL_VIOLATION_PROCEDURE)
- EXCEPTION_LIST: Forma la llista d'excepcions als processos de la Blacklist. Està ideat per no afectar a certes execucions amb paràmetres específics. Per defecte és buida. Un exemple pot ser:
    - **EXCEPTION_LIST=("pcmanfm --desktop --profile LXDE")**

La última part consta dels paràmetres de Idle Watchdog:

- IDLE_KILL_LIST: Conté la llista de processos que finalitzaran quan arribi el temps d'inactivitat especificat. Per defecte és buida. Un exemple pot ser:
    - **IDLE_KILL_LIST=("chromium-browse")**
- MINUTS_IN_IDLE_STATE: Son els minuts d'inactivitat de l'usuari. El valor mínim es 0.1 (6 segons). I per defecte es 1.
    - **MINUTS_IN_IDLE_STATE=1**
- SECONDS_TO_CLOSE: Son els segons d'espera del programa per donar el procés per tancat. El valor mínim es 0. I per defecte es 5.
    - **SECONDS_TO_CLOSE=5**
- TIMEFILE_ENABLED: És el paràmetre per activar la creació i enregistrament de l'arxiu de temps. Per defecte està desactivat:
    - **TIMEFILE_ENABLED=false**
- TIMEFILENAME: Nom l'arxiu de temps. Per defecte es "idle-watchdog.time"
    - **TIMEFILENAME="idle-watchdog.time"**

Contingut del programa

activity-watchdog.sh

```
#!/bin/bash
export DISPLAY=:0

#########################################################################
# Created by: Nahúm Manuel Martín Vegas                                 #
# URL: https://github.com/nadamas2000/activity-watchdog                 #
# File: activity-watchdog.sh                                            #
# Version: V1                                                           #
# Date: 04-dec-2019                                                     #
# Developed in: Facultat d´informàtica de Barcelona (FIB)               #
#               Universitat Politécnica de Catalunya (UPC)              #
#                                                                       #
# Project name: Activity-watchdog for Raspberry Pi (Raspbian linux)     #
# Project files:        activity-watchdog.sh                            #
#                       activity-watchdog.conf                          #
#                       software-watchdog.shlib                         #
#                       idle-watchdog.shlib                             #
#########################################################################


SOURCE=${BASH_SOURCE[0]}
EXEDIR=$(dirname $SOURCE)

# Default parameters. This parameters are changed by activity-watchdog.conf
USER="username"
LOGFILEDIR=$EXEDIR
LOGFILENAME="activity-watchdog.log"
LOGFILE_BY_EXECUTION=true
WAITING_FOR_PROC="Xorg"
FIRST_WAITING=20
OBS_INTERVAL=1
FORCE_LIST_WATCHDOG_ENABLED=true
BLACKLIST_WATCHDOG_ENABLED=true
IDLE_WATCHDOG_ENABLED=true
declare -a FORCE_LIST_PROC BLACKLIST_PROC EXCEPTION_LIST IDLE_KILL_LIST
GENERAL_VIOLATION_PROCEDURE=""
MINUTS_IN_IDLE_STATE=1
SECONDS_TO_CLOSE=5
TIMEFILE_ENABLED=false
TIMEFILENAME="idle-watchdog.time"

# Load config file
CONFIGFILE="$EXEDIR/activity-watchdog.conf"
if [ ! -f "$CONFIGFILE" ]; then
        echo "No es troba el fitxer de configuració"
        exit 1
else
        source $CONFIGFILE
fi

LOGFILE="$LOGFILEDIR/$LOGFILENAME"
```

```
# Force parameters to minimum value.
if [[ $FIRST_WAITING -lt 0 ]]; then
        FIRST_WATING=0
fi
if [[ $OBS_INTERVAL -le 0 ]]; then
        OBS_INTERVAL=1
fi
if [[ $MINUTS_IN_IDLE_STATE -lt 1 ]]; then
        $MINUTS_IN_IDLE_STATE=1
fi
if [[ $SECONDS_TO_CLOSE -lt 0 ]]; then
        $SECONDS_TO_CLOSE=0
Fi


##################### FUNCTIONS AND SHLIBs ############################

# Function to add info to logfile
msgToLog () {
        $(echo $(date)" "$1 >> $LOGFILE)
}

# Load software-watchdog.shlib
SOFTWAREWATCHDOG="$EXEDIR/software-watchdog.shlib"
if [ ! -f "$SOFTWAREWATCHDOG" ]; then
        echo "software-watchdog file not found"
        exit 1
else
        source $SOFTWAREWATCHDOG
fi


# Load idle-watchdog.shlib
IDLEWATCHDOG="$EXEDIR/idle-watchdog.shlib"
if [ ! -f "$IDLEWATCHDOG" ]; then
        echo "idle-watchdog file not found"
        exit 1
else
        source $IDLEWATCHDOG
fi



######################### MAIN ############################
main() {
        if $LOG_FILE_BY_EXECUTION; then
                echo $(date)" Creating new logfile..." > $LOGFILE
        else
                echo $(date)" Starting activity-watchdog..." >> $LOGFILE
        fi

        msgToLog "Waiting for the first execution of $WAITING_FOR_PROC ..."
        declare -a PIDX                 # declare array PIDX
        PIDX=($(pgrep $WAITING_FOR_PROC))       # Obtain PIDs with
$WAITING_FOR_PROC name to array
        # Waiting process
        while [ -z "${PIDX[0]}" ]; do
                sleep1
                PIDX=($(pgrep $WAITING_FOR_PROC))
```

```
        done
        msgToLog "$WAITING_FOR_PROC detected with PID: ${PIDX[0]}"
        msgToLog "Waiting $FIRST_WAITING"
        sleep $FIRST_WAITING            # Time to load the process

        ##################### PERSISTENT ACTIVITY ######################
        msgToLog "Initializing watchdog (while-true)"
        while [ true ]; do
                sleep $OBS_INTERVAL
                softwareWatchdog
                if $IDLE_WATCHDOG_ENABLED; then
                        idleWatchdog
                fi
        done
}

main "$@"
```

software-watchdog.shlib

```
#########################################################################
# Created by: Nahúm Manuel Martín Vegas                                 #
# URL: https://github.com/nadamas2000/activity-watchdog                 #
# File: software-watchdog.shlib                                         #
# Version: V1                                                           #
# Date: 04-dec-2019                                                     #
# Developed in: Facultat d´informàtica de Barcelona (FIB)               #
#               Universitat Politécnica de Catalunya (UPC)              #
#                                                                       #
# Project name: Activity-watchdog for Raspberry Pi (Raspbian linux)     #
# Project files:        activity-watchdog.sh                            #
#                       activity-watchdog.conf                          #
#                       software-watchdog.shlib                         #
#                       idle-watchdog.shlib                             #
#########################################################################


# Function that inspect the executed programs by the user and forces to
# execute all software in configuration parameter (FORCE_LIST_PROC)
watchForcedList () {
        local i=0
        while [[ i -lt ${#FORCE_LIST_PROC[@]} ]];
        do
                declare -a PID0
                local PROC="${FORCE_LIST_PROC[i]}"
                PID0=($(pgrep -u $USER $PROC | cut -d " " -f1))
                if [[ -z "${PID0[0]}" ]]; then
                        msgToLog "$PROC is not running"
                        i=$i+1
                        local EPROC=${FORCE_LIST_PROC[i]}
                        msgToLog "Executing $EPROC"
                        $EPROC
                        sleep 2

                        msgToLog "$PROC restarted"
                        i=$i+1
                else
                        i=$i+2
                fi

        done
}


# Function that inspect the executed programs by the user and forces
# to close all software in configuration parameters (BLACKLIST_PROC,
# EXCEPTION_LIST)
watchBlacklist () {
        declare -a EXCEPTION_PIDs PS PIDs EXEC
        PIDs=($(pgrep -u $USER -a | cut -d " " -f1))
        PGREP=$(pgrep -u $USER -a | cut -d " " -f2-10)
        PGREP=$(echo ${PGREP// /ç})     # Replace scapaces by 'ç'
        EXEC=($(echo $PGREP))           # Convert text to strings array

        local i=0
```

```
        while [[ i -lt ${#EXCEPTION_LIST[@]} ]]; do
                local PROC=${EXCEPTION_LIST[i]}
                e=0
                while [[ e -lt "${#PIDs[@]}" ]]; do
                        EXECPROC=$(echo ${EXEC[e]} | tr -s 'ç' ' ')     #
Replace 'ç' by spaces
                        if [[ $EXECPROC = $PROC ]]; then
                                EXCEPTION_PIDs+=${PIDs[e]}
                        fi
                        e=$e+1
                done
                i=$i+1
        done

        declare -a BLACKLIST_PIDs PROCEDs
        i=0
        while [[ i -lt ${#BLACKLIST_PROC[@]} ]]; do
                local PROC="${BLACKLIST_PROC[i]}"
                BLACKLIST_PIDs=($(pgrep -u $USER $PROC))
                for d in "${EXCEPTION_PIDs[@]}"; do
                        BLACKLIST_PIDs=( ${BLACKLIST_PIDs[@]/$d} )
                done

                i=$i+1
                if [[ ${#BLACKLIST_PIDs[@]} -gt 0 && ${BLACKLIST_PIDs[0]} -
ne 0 ]]; then
                        for k in "${BLACKLIST_PIDs[@]}"; do
                                msgToLog "$PROC detected with PID: $k"
                                sudo pkill $k
                        done
                        local VPROC="${BLACKLIST_PROC[i]}"
                        msgToLog "Executing $VPROC"
                        $VPROC >> $LOGFILE
                        sleep 2
                        msgToLog "Blacklist violation procedure executed"
                fi
                i=$i+1
        done
}


# The main function to execute the different procedures depending on
# the activation configuration specified within the configuration file
# (FORCE_LIST_WATCHDOG_ENABLED, BLACKLIST_WATCHDOG_ENABLED)
softwareWatchdog () {
        if $FORCE_LIST_WATCHDOG_ENABLED; then
                watchForcedList
        fi
        if $BLACKLIST_WATCHDOG_ENABLED; then
                watchBlacklist
        fi
}
```

idle-watchdog.shlib

```
#########################################################################
# Created by: Nahúm Manuel Martín Vegas                                 #
# URL: https://github.com/nadamas2000/activity-watchdog                 #
# File: idle-watchdog.shlib                                             #
# Version: V1                                                           #
# Date: 04-dec-2019                                                     #
# Developed in: Facultat d´informàtica de Barcelona (FIB)               #
#               Universitat Politécnica de Catalunya (UPC)              #
#                                                                       #
# Project name: Activity-watchdog for Raspberry Pi (Raspbian linux)     #
# Project files:        activity-watchdog.sh                            #
#                       activity-watchdog.conf                          #
#                       software-watchdog.shlib                         #
#                       idle-watchdog.shlib                             #
#########################################################################


# Timefile with local execution address
TIMEFILE=$EXEDIR"/"$TIMEFILENAME

############################ FUNCTIONS ############################

# Message to Timefile
msgToTimefile () {
        $(echo $1 >> $TIMEFILE)

}

# Data that contains Timefile
timefileWriter () {
        if $TIMEFILE_ENABLED; then
                let isec=$(( ($1 /        1000 ) % 60 ))
                let imin=$(( ($1 /    (60*1000)) % 60 ))
                let ihor=$(( ($1 / (60*60*1000)) % 60 ))

                date > $TIMEFILE        # New Timefile
                msgToTimefile "Idle time: "$1" = "$ihor"h "$imin"m
"$isec"s"
                msgToTimefile "Idle max: $MINUTS_IN_IDLE_STATE m"
                msgToTimefile "Idle stop: $MINUTS_IN_IDLE_STATE m
$SECONDS_TO_CLOSE s"

        fi
}

# Add info to timefile that contains the idle state.
addInfoToTimefile () {
        if $TIMEFILE_ENABLED; then
                let msgcase=$1
                if [ $msgcase -eq 1 ]; then
                        msgToTimefile "Watchdog up (idle < idle_stop):
$idle < $idle_stop"
                fi
                if [ $msgcase -eq 2 ]; then
```

```
                        msgToTimefile "Watchdog restarting
$PROC_NAME_TO_KILL (idle > idle_max): $idl$
                fi
                if [ $msgcase -eq 3 ]; then
                        msgToTimefile "Watchdog down (idle > idle_stop):
$idle > $idle_stop"
                fi
        fi
}

# Killing process list function
killProc () {
        msgToLog "Idle time exceded: $idle"
        for i in "${IDLE_KILL_LIST[@]}"
        do
                local PIDK=($(pgrep -u $USER $i))
                if [ -n "$PIDK" ]; then
                        killall -9 $i
                        msgToLog "Command to kill $i executed"
                fi
        done
        sleep $SECONDS_TO_CLOSE
}


######################### MAIN FUNCTION ############################
idleWatchdog () {
        export DISPLAY=:0
        idle=$(sudo -u pi xprintidle)
        idle_max=$(( $MINUTS_IN_IDLE_STATE * 60 * 1000 ))
        idle_stop=$(( $idle_max + ($SECONDS_TO_CLOSE * 1000) ))
        timefileWriter $idle

        if [ $idle -lt $idle_stop ];
        then
                $(addInfoToTimefile 1)
                if [[ $idle -gt $idle_max ]];
                then
                        $(addInfoToTimefile 2)
                        $(killProc)
                fi
        else
                $(addInfoToTimefile 3)
        fi

}
```

activity-watchdog.conf

```
########################################################################
# Created by: Nahúm Manuel Martín Vegas                                #
# URL: https://github.com/nadamas2000/activity-watchdog                #
# File: activity-watchdog.conf                                         #
# Version: V1                                                          #
# Date: 03-dec-2019                                                    #
# Developed in: Facultat d´informàtica de Barcelona (FIB)              #
#               Universitat Politécnica de Catalunya (UPC)             #
#                                                                      #
# Project name: Activity-watchdog for Raspberry Pi (Raspbian linux)    #
# Project files:        activity-watchdog.sh                           #
#                       activity-watchdog.conf                         #
#                       software-watchdog.shlib                        #
#                       idle-watchdog.shlib                            #
# Dependecies:  bash                                                   #
#               procps                                                 #
#               coreutils                                              #
#               xprintidle (only for idle-watchdog function)           #
########################################################################


######################### GENERAL CONFIGURATION ############################

# User to observe. Only one user. To use this software for more user
# you can use different instances in different execution folders.
USER="pi"

# Logfile address. This address can't contains a final character "/".
# $EXEDIR (execution directory) by default to store the logfile in the
# same folder of the software.
LOGFILEDIR=$EXEDIR

# Logfile name.
LOGFILENAME="activity-watchdog.log"

# New Logfile by execution. The oldest logfile is rewrited.
LOG_FILE_BY_EXECUTION=true

# Process that the watchdogs waiting for.
WAITING_FOR_PROC="Xorg"

# First timeout to load process in WAITING_FOR_PROC parameter (seconds).
# Minimum value is 0. Default value 20.
FIRST_WAITING=20

# Observation interval (seconds). Minimum 1.
OBS_INTERVAL=1

# Idle watchdog activation.
IDLE_WATCHDOG_ENABLED=true

# Force-list watchdog activation.
FORCE_LIST_WATCHDOG_ENABLED=true

# Blacklist watchdog activation.
```

```
BLACKLIST_WATCHDOG_ENABLED=true


################ FORCE-LIST WATCHDOG CONFIGURATION ###############

# List of pairs "process name" - "action to execute" that we force on.
# Always in pairs. This parameter is a shell array of strings.
# The process name can't be contain parameters.
FORCE_LIST_PROC=("chromium-browse"      "/home/soft/bin/start-chrome-
pi.sh")


############### BLACKLIST WATCHDOG CONFIGURATION #################

# General script or execution command by Blacklist violation.
# Empty by default.
GENERAL_VIOLATION_PROCEDURE="/home/soft/bin/reiniciarX.sh"

# List of pairs "process name" - "action to execute" that we need to
# preserve not executed.
# Always in pairs. This parameter is a shell array of strings.
# The process name can't be contain execution parameters.
# $GENERAL_VIOLATION_PROCEDURE by default. You can use other personal
# scripts.
BLACKLIST_PROC=("pcmanfm"       $GENERAL_VIOLATION_PROCEDURE \
                "vi"            $GENERAL_VIOLATION_PROCEDURE \
                "vim"           $GENERAL_VIOLATION_PROCEDURE \
                "nano"          $GENERAL_VIOLATION_PROCEDURE \
                "lxterminal"    $GENERAL_VIOLATION_PROCEDURE \
                "xterm"         $GENERAL_VIOLATION_PROCEDURE \
                "uxterm"        $GENERAL_VIOLATION_PROCEDURE)

# Exception process list specified with or without parameters in a single
# string by exception. You can use "pgrep -a" to obtain specified
# parameters by process.
EXCEPTION_LIST=("pcmanfm --desktop --profile LXDE")


###################### IDLE WATCHDOG CONFIGURATION ######################

# Process list to kill when idle time ends.
IDLE_KILL_LIST=("chromium-browse")

# Time limit in minutes to kill all the processes.
# Minimum and default value 1 minute.
MINUTS_IN_IDLE_STATE=1

# Margin time to close all processes in the list.
# Minimum value 0. Default value 5.
SECONDS_TO_CLOSE=5

# Timefile activation. Default value false.
TIMEFILE_ENABLED=false

# File to watch the current idle time and other info.
TIMEFILENAME="idle-watchdog.time"
```