Mint update on wakeup

v2 is available here: https://community.linuxmint.com/tutorial/view/2429

Prerequisites:

```
sudo apt-get install at libnotify-bin
```

mintupdate-on-wakeup.sh

```
#!/bin/bash
ARGS=$1; SCRIPTDIR=${0%/*}; SCRIPTNAME=${0##*/};
SCRIPTNAME=${SCRIPTNAME#*_}
#######################################################################
#########
#
# mintupdate-on-wakeup by tux@enkidu.eu
#
# (based on unattended-upgrades-alternative 1.1 by tux@enkidu.eu)
#
# 28.12.2018: v1.0 - initial release (Mint 18 only)
# 30.04.2021: v2.0 - added support for mintupdate-cli
#
#######################################################################
#########
AUTHOR="tux@enkidu.eu"; VER=2.0

#######################################################################
########
#
# This is a simple replacement for "unattended-upgrades".
# It is NOT thought for server usage!
# It is thought for e.g. users with a notebook, where normal usage is
#   open notebook = resume/wakeup
#  close notebook = hibernate/suspend
#
# Script is executed after each wakeup and update done 2-3 times a day.
# (Not at every resume, only if first digit of 2 digit hour changes.
#  E.g. 0x (00, 01, 02, ...),  1x, 2x)
#
#######################################################################
#########

#######################################################################
#########
#
#  >>> Use it on your own risk! <<<<
#
#######################################################################
```

```
#########

##############################################################################
#
# Installation:
# -------------
#
# Place this script in $HOME/bin
#
# > chmod 755 $HOME/bin/mintupdate-on-wakeup
# > sudo ln -s $HOME/bin/mintupdate-on-wakeup /lib/systemd/system-
sleep/
#
#
# If you want to check for updates after login, you have to add
#
#   sudo $HOME/bin/mintupdate-on-wakeup login
#
# to your
#
#   $HOME/.profile
#
# AND edit your sudoers config accordingly that it can be run without
pw!
#
#
# IMPORTANT: Please see also configuration below!
#
##############################################################################

##############################################################################
#
# Usage:
# ------
#
# For testing purposes you can invoke it like
#
# > sudo ./mintupdate-on-wakeup post
#
#
# To clean logs, index, tmp, start it with
#
# > sudo ./mintupdate-on-wakeup cleanup
#
##############################################################################

#######################################################################
```

```
#########
#
# See /var/log/mintupdate-on-wakeup.log for log!
#
########################################################################
#########

#=======================================================================
=========
#
# User configuration
#

# If you are still on LM 18, use:
# Your mintupdate-tool settings (-r -y is added by default)
# See "mintupdate-tool --help" for details
MUTOPTS="-k -s -l 1234"

# From LM 19 on, use:
# Your mintupdate-cli settings (-r -y is added by default)
# See "mintupdate-cli --help" for details
MUCOPTS=""

# if mintupdate-cli/tool returns with error, automatically open Mint
Update GUI?
OPENUPDGUIONERR=true  # or false

# Delay in minutes the script waits before mintupdate-cli/tool is
started
UPDATEDELAY=2         # minimum = 1!


# Desktop notifications
#
# Prerequisite: libnotify-bin
# If not installed it will simply not notify! :)

# Level 0:  disable desktop notifications
#       1:  errors
#       2:  warning
#       3:  info
#       4:  verbose
#       5:  debug
#
# For normal usage, I suggest to use 2.
NOTIFY_LEVEL=5


# The following variables only need to be changed if the values
returend
# do not fit your needs!
# Icons used for notification. Change only if they do not fit or do not
exist
NOTIFY_ICON_AVAIL="/usr/share/icons/hicolor/24x24/status/mintupdate-
updates-available.png"
NOTIFY_ICON_CHECK="/usr/share/icons/hicolor/24x24/status/mintupdate-
checking.png"
NOTIFY_ICON_ERROR="/usr/share/icons/hicolor/24x24/status/mintupdate-
error.png"
NOTIFY_ICON_INSTALL="/usr/share/icons/hicolor/24x24/status/mintupdate-
installing.png"
#NOTIFY_ICON_AVAIL="/usr/share/icons/hicolor/48x48/status/mintupdate-
updates-available.png"
#NOTIFY_ICON_CHECK="/usr/share/icons/hicolor/48x48/status/mintupdate-
checking.png"
#NOTIFY_ICON_ERROR="/usr/share/icons/hicolor/48x48/status/mintupdate-
error.png"
#NOTIFY_ICON_INSTALL="/usr/share/icons/hicolor/48x48/status/mintupdate-
installing.png"


# Here you may manually enter the value you get in a normal terminal
when typing
# > echo $DISPLAY
# BUT ONLY change it, if the display could not properly be detected!
DESKTOP_DISPLAY=":$( ls /tmp/.X11-unix/* | head -1 | sed 's:/tmp/.X11-
unix/X::' )"

# What is the name of the desktop user you are working with?
# Some commands need to be invoked inside your user context or it does
not work.
# BUT ONLY change it, if the user could not properly be detected!
DESKTOP_USER=$( who | grep '('$DESKTOP_DISPLAY')' | head -1 | awk
'{print $1}' )

#
# End of user configuration
#
#=======================================================================
=========


#######################################################################
#########
#######################################################################
#########
##
##  DO NOT EDIT BELOW IF YOU DO NOT REALLY KNOW WHAT YOU ARE DOING!!!
##
#######################################################################
```

```
#########
########################################################################
#########


[ $USER != root ] && {
    echo "
$SCRIPTNAME: This script must be run as root!

Usage: sudo $SCRIPTNAME < <post|resume|thaw|login> |
<pre|hibernate|suspend> | clean >
"
    exit 1
}

log() {
    # $1: log message level
    #  2: text
    #  3: additional text

    DT=$( date +'%Y-%m-%d %H-%M-%S' )
    printf "%s | %5d | %d - %s\n" "$DT" $$ $1  "$2"
    [ ! -z "$3" ] && printf "%s | %5d | %s   %s\n" "$DT" $$ ' ' "$3"


    # send desktop notification
    [ -z "$NOTIFY_SEND" ]   && return
    (( NOTIFY_LEVEL < $1 )) && return

    case $1 in
        1)  ICO=$NOTIFY_ICON_ERROR;   URG=critical;;
        2)  ICO=$NOTIFY_ICON_INSTALL; URG=normal;;
        3)  ICO=$NOTIFY_ICON_AVAIL;   URG=low;;
        4)  ICO=$NOTIFY_ICON_CHECK;   URG=low;;
        5)  ICO=$NOTIFY_ICON_CHECK;   URG=low;;
    esac

    IC=''
    [ -r $ICO ] && IC="-i $ICO" || echo >&2 "ERROR: unable to read icon
'$ICO'!"

    BODY="$2"
    [ ! -z "$3" ] && BODY+="\n$3"

    TITLE=$( printf "%-70s" "<< $SCRIPTNAME >>" )

    # notify-send -a $SCRIPTNAME $IC -u $URG "$TITLE" "$BODY"
    su -c "$NOTIFY_SEND -a $SCRIPTNAME $IC -u $URG '$TITLE' '$BODY'"
$DESKTOP_USER
}
```

```
quit() {
    log 5 "*** $SCRIPTNAME ended ***"
    echo
    exit $1
}

tfunc() {
    type $1 | sed -n '2,$p'
}


# some presets
LOG_FILE=/var/log/$SCRIPTNAME.log

# what mintupdate to use?
which mintupdate-cli 2>&1 >/dev/nul && {
    MUCMD=mintupdate-cli
    UPDCMD="$MUCMD -r -y $MUCOPTS upgrade"
} || {
    MUCMD=mintupdate-tool
    UPDCMD="$MUCMD -r -y $MUTOPTS upgrade"
}

CHECKUPDRUNNING="ps -ef | fgrep -v grep | egrep 'apt-get|dpkg|$MUCMD' |
wc -l"
NOTIFY_SEND=${NOTIFY_SEND:-notify-send}
which $NOTIFY_SEND >/dev/null 2>&1 || NOTIFY_SEND=''

# get DBUS of user (it is tricky and might not work for all...)
[ -z "$DBUS_SESSION_BUS_ADDRESS" ] && {
    DUID=$( pgrep -l "(cinnamon|gnome|kde|mate|xfce)" | fgrep "-sessio"
| awk '{ print $1 }' )
    DSBA=$( grep -z '^DBUS_SESSION_BUS_ADDRESS' /proc/$DUID/environ )
    export DBUS_SESSION_BUS_ADDRESS=${DSBA#*=}
} 2>/dev/null

# check DISPLAY
[ -z "$DISPLAY" ] && \
    export DISPLAY="$DESKTOP_DISPLAY"

LASTRUN_INDEX=/var/tmp/$SCRIPTNAME.lastrun
TMP_FILE=/var/tmp/$SCRIPTNAME.tmp


# redirect all output to logfile
exec >>$LOG_FILE
exec 2>&1


log 5 "*** $SCRIPTNAME v$VER by $AUTHOR started ***" "[ \$1='$ARGS';
USER=$DESKTOP_USER; DISPLAY=$DISPLAY}; DSBA=$DBUS_SESSION_BUS_ADDRESS
```

```
]"

# is "at" installed?
which at >/dev/null 2>&1 || {
    log 1 'ERROR: "at" command not found! Use "sudo apt-get install
at"!'
    quit 2
}

[ -z "$DBUS_SESSION_BUS_ADDRESS" ] && \
    log 2 "WARNING: DBUS_SESSION_BUS_ADDRESS could not be detected!
Desktop notifications might not work!" \
          "Please contact $AUTHOR and provide infos about your
environment to fix this problem!"

case "$ARGS" in
    post|resume|thaw|login)
        log 4 "## Wakeup ($ARGS)"

        # read last run
        LASTRUN=`cat $LASTRUN_INDEX 2>/dev/null || echo 0`
        NOW=`date "+%Y%m%d%H" | cut -b 1-9`

        # Skip, if update already in progress!
        if (( `eval "$CHECKUPDRUNNING"` )); then
            log 3 "Skipped! (Update in progress!)"
        # Skip update if in same hour range
        elif (( LASTRUN == NOW )); then
            log 3 "Skipped! Already done a while ago." "[ Last $LASTRUN
== Now $NOW ]"
        # Check & install updates
        else
            # run this part in background, otherwise it will conflict
with other wakeup scripts!
            cat > $TMP_FILE <<__EOT__
#!/bin/bash
trap "" 1 2 3 15
LOG_FILE=$LOG_FILE
SCRIPTNAME=$SCRIPTNAME
exec >>$LOG_FILE
exec 2>&1
export DBUS_SESSION_BUS_ADDRESS=$DBUS_SESSION_BUS_ADDRESS
export DISPLAY="$DESKTOP_DISPLAY"
`set | egrep '^DISPLAY|^XAUTHORITY' | while read L; do echo "export
$L"; done`
`set | egrep '^DESKTOP_|^NOTIFY_'`
`tfunc log`
`tfunc quit`
            log 5 "Background process initiated from PID \$1" "[
DESKTOP_USER=$DESKTOP_USER; DISPLAY=$DISPLAY ]"
```

```
            # check internet connection first
            log 4 "Checking connectivity..."
            I=0
            while true
            do
                (( I > 20 )) && {
                    log 1 "ERROR: could not detect internet
connectivity!"
                    quit 1
                }

                let I=I+1

                DG=\$( ip r | fgrep default | awk '{ print \$3; }' )
                [ -z "\$DG" ] && {
                    log 5 "> No default gateway found, waiting 10sec
for interface..."
                    sleep 10
                    continue
                }

                ping -c 1 -q -W 3 \$DG || {
                    log 5 "> default gateway '\$DG' is not reachable,
waiting 10sec to try again..."
                    sleep 10
                    continue
                }

                ping -c 1 -q -W 3 www.google.com || {
                    log 5 "> www.google.com is not reachable, waiting
10sec to try again..."
                    sleep 10
                    continue
                }

                break
            done
            log 4 ">> Ok!"

            log 3 "Starting Mint Update!" "[ Last $LASTRUN != Now $NOW
]"
            $UPDCMD \
                &&  log 4 "> $MUCMD finished with no error" \
                || {
                    log 1 "> $MUCMD returned error \$?!" ">> Check
$LOG_FILE for details & retry using mintupdate gui!"
                    [ "$OPENUPDGUIONERR" == "true" ] \
                        && echo "su -c 'export
DISPLAY=$DESKTOP_DISPLAY; mintupdate'" $DESKTOP_USER | at now
                   }
            echo >$LASTRUN_INDEX "$NOW"
            quit 0
__EOT__
            log 5 "Starting update background process in $UPDATEDELAY
min..." "[ $TMP_FILE ]"
            chmod +x $TMP_FILE
            echo $TMP_FILE $$ | at -M now + $UPDATEDELAY min
            # Using at was the only way I could find that my child
process was not killed
            # (nor disown & or hohup & did survive - dunno why!?) or
caused strange
            # side effects. Using at everything works perfect! :)
        fi
        ;;

    pre|hibernate|suspend)
        log 4 "## Suspend ($ARGS)"
        # if going in hibernate/suspend just wait a moment if an update
is running.
        # But even if, it should be continued after resume...
        # So no complicated hibernate handling is required.

        let I=0
        while (( `eval "$CHECKUPDRUNNING"` ))
        do
            (( I == 10 )) && break
            ((    ! I   )) && log 2 "> Running updates detected!"
"Waiting max. 100sec to go to sleep ($ARGS)..."
            ((      I   )) && log 4 "> ... $I"
            sleep 10
            let I=I+1
        done
        (( ! I )) && log 5 "> No running updates detected."
        quit 0
        ;;

    cleanup)
        rm -f $LASTRUN_INDEX $LOG_FILE $TMP_FILE >/dev/null 2>&1
        log 4 ">> Cleanup finished <<"
        ;;

    *)  log 1 "Mode '$ARGS' ist not supported!" "Please report to
$AUTHOR if it was not a cmdline try by yourself!"
        quit 1
        ;;
esac
```

Version 1 for reference:

mintupdate-on-wakeup.sh

```
#!/bin/bash
ARGS=$1; SCRIPTNAME=${0##*/}; SCRIPTNAME=${SCRIPTNAME#*_}
######################################################################
#########
#
# mintupdate-on-wakeup by tux@enkidu.eu
#
# (based on unattended-upgrades-alternative 1.1 by tux@enkidu.eu)
#
# 28.12.2018: v1.0 - initial release
#
######################################################################
#########
AUTHOR="tux@enkidu.eu"; VER=1.0

######################################################################
########
#
# This is a simple replacement for "unattended-upgrades".
# It is NOT thought for server usage!
# It is thought for e.g. users with a notebook, where normal usage is
#   open notebook = resume/wakeup
#  close notebook = hibernate/suspend
#
# Script is executed after each wakeup and update done 2-3 times a day.
# (Not at every resume, only if first digit of 2 digit hour changes.
#  E.g. 0x (00, 01, 02, ...),  1x, 2x)
#
######################################################################
#########

######################################################################
#########
#
#  >>> Use it on your own risk! <<<<
#
######################################################################
#########

######################################################################
#########
#
# Installation:
# -------------
#
# Place this script in $HOME/bin
#
# > chmod 755 $HOME/bin/mintupdate-on-wakeup
# > sudo ln -s $HOME/bin/mintupdate-on-wakeup /lib/systemd/system-
sleep/
#
```

```
#
# If you want to check for updates after login, you have to add
#
#   sudo $HOME/bin/mintupdate-on-wakeup login
#
# to your
#
#   $HOME/.profile
#
# AND edit your sudoers config accordingly that it can be run without
pw!
#
#
# IMPORTANT: Please see also configuration below!
#
########################################################################
#########

########################################################################
#########
#
# Usage:
# ------
#
# For testing purposes you can invoke it like
#
# > sudo ./mintupdate-on-wakeup post
#
#
# To clean logs, index, tmp, start it with
#
# > sudo ./mintupdate-on-wakeup cleanup
#
########################################################################
#########

########################################################################
#########
#
# See /var/log/mintupdate-on-wakeup.log for log!
#
########################################################################
#########

#=======================================================================
=========
#
# User configuration
#

# Your mintupdate-tool settings (-r -y is added by default)
# See "mintupdate-tool --help" for details
MUTOPTS="-k -s -l 1234"

# if mintupdate-tool returns with error, automatically open Mint Update
GUI?
OPENUPDGUIONERR=true  # or false

# Delay in minutes the script waits before mintupdate-tool is started
UPDATEDELAY=2         # minimum = 1!


# Desktop notifications
#
# Prerequisite: libnotify-bin
# If not installed it will simply not notify! :)

# Level 0:  disable desktop notifications
#       1:  errors
#       2:  warning
#       3:  info
#       4:  verbose
#       5:  debug
#
# For normal usage, I suggest to use 2.
NOTIFY_LEVEL=5


# The following variables only need to be changed if the values
returend
# do not fit your needs!


# Icons used for notification. Change only if they do not fit or do not
exist
NOTIFY_ICON_AVAIL="/usr/share/icons/hicolor/24x24/status/mintupdate-
updates-available.png"
NOTIFY_ICON_CHECK="/usr/share/icons/hicolor/24x24/status/mintupdate-
checking.png"
NOTIFY_ICON_ERROR="/usr/share/icons/hicolor/24x24/status/mintupdate-
error.png"
NOTIFY_ICON_INSTALL="/usr/share/icons/hicolor/24x24/status/mintupdate-
installing.png"
#NOTIFY_ICON_AVAIL="/usr/share/icons/hicolor/48x48/status/mintupdate-
updates-available.png"
#NOTIFY_ICON_CHECK="/usr/share/icons/hicolor/48x48/status/mintupdate-
checking.png"
#NOTIFY_ICON_ERROR="/usr/share/icons/hicolor/48x48/status/mintupdate-
error.png"
#NOTIFY_ICON_INSTALL="/usr/share/icons/hicolor/48x48/status/mintupdate-
installing.png"
```

```
# Here you may manually enter the value you get in a normal terminal
when typing
# > echo $DISPLAY
# BUT ONLY change it, if the display could not properly be detected!
DESKTOP_DISPLAY=":$( ls /tmp/.X11-unix/* | head -1 | sed 's:/tmp/.X11-
unix/X::' )"

# What is the name of the desktop user you are working with?
# Some commands need to be invoked inside your user context or it does
not work.
# BUT ONLY change it, if the user could not properly be detected!
DESKTOP_USER=$( who | grep '('$DESKTOP_DISPLAY')' | head -1 | awk
'{print $1}' )

#
# End of user configuration
#
#=======================================================================
=========


#######################################################################
#########
#######################################################################
#########
##
##  DO NOT EDIT BELOW IF YOU DO NOT REALLY KNOW WHAT YOU ARE DOINT!!!
##
#######################################################################
#########
#######################################################################
#########


[ $USER != root ] && {
    echo "
$SCRIPTNAME: This script must be run as root!

Usage: sudo $SCRIPTNAME < <post|resume|thaw|login> |
<pre|hibernate|suspend> | clean >
"
    exit 1
}

log() {
    # $1: log message level
    #  2: text
    #  3: additional text

    DT=$( date +'%Y-%m-%d %H-%M-%S' )
    printf "%s | %5d | %d - %s\n" "$DT" $$ $1  "$2"
    [ ! -z "$3" ] && printf "%s | %5d | %s   %s\n" "$DT" $$ ' ' "$3"


    # send desktop notification
    [ -z "$NOTIFY_SEND" ]   && return
    (( NOTIFY_LEVEL < $1 )) && return

    case $1 in
        1)  ICO=$NOTIFY_ICON_ERROR;   URG=critical;;
        2)  ICO=$NOTIFY_ICON_INSTALL; URG=normal;;
        3)  ICO=$NOTIFY_ICON_AVAIL;   URG=low;;
        4)  ICO=$NOTIFY_ICON_CHECK;   URG=low;;
        5)  ICO=$NOTIFY_ICON_CHECK;   URG=low;;
    esac

    IC=''
    [ -r $ICO ] && IC="-i $ICO" || echo >&2 "ERROR: unable to read icon
'$ICO'!"

    BODY="$2"
    [ ! -z "$3" ] && BODY+="\n$3"

    TITLE=$( printf "%-70s" "<< $SCRIPTNAME >>" )

    # notify-send -a $SCRIPTNAME $IC -u $URG "$TITLE" "$BODY"
    su -c "$NOTIFY_SEND -a $SCRIPTNAME $IC -u $URG '$TITLE' '$BODY'"
$DESKTOP_USER
}

quit() {
    log 5 "*** $SCRIPTNAME ended ***"
    echo
    exit $1
}

tfunc() {
    type $1 | sed -n '2,$p'
}


# some presets
LOG_FILE=/var/log/$SCRIPTNAME.log

CHECKUPDRUNNING="ps -ef | fgrep -v grep | egrep 'apt-
get|dpkg|mintupdate-tool' | wc -l"
NOTIFY_SEND=${NOTIFY_SEND:-notify-send}
which $NOTIFY_SEND >/dev/null 2>&1 || NOTIFY_SEND=''

# get DBUS of user (it is tricky and might not work for all...)
[ -z "$DBUS_SESSION_BUS_ADDRESS" ] && {
    DUID=$( pgrep -l "(cinnamon|gnome|kde|mate|xfce)" | fgrep "-sessio"
| awk '{ print $1 }' )
    DSBA=$( grep -z "^DBUS_SESSION_BUS_ADDRESS" /proc/$DUID/environ )
    export DBUS_SESSION_BUS_ADDRESS=${DSBA#*=}
}

# check DISPLAY
[ -z "$DISPLAY" ] && \
    export DISPLAY="$DESKTOP_DISPLAY"

LASTRUN_INDEX=/var/tmp/$SCRIPTNAME.lastrun
TMP_FILE=/var/tmp/$SCRIPTNAME.tmp


# redirect all output to logfile
exec >>$LOG_FILE
exec 2>&1


log 5 "*** $SCRIPTNAME v$VER by $AUTHOR started ***" "[ \$1='$ARGS';
USER=$DESKTOP_USER; DISPLAY=$DISPLAY}; DSBA=$DBUS_SESSION_BUS_ADDRESS
]"

[ -z "$DBUS_SESSION_BUS_ADDRESS" ] && \
    log 2 "WARNING: DBUS_SESSION_BUS_ADDRESS could not be detected!
Desktop notifications might not work!" \
          "Please contact $AUTHOR and provide infos about your
environment to fix this problem!"

case "$ARGS" in
    post|resume|thaw|login)
        log 4 "## Wakeup ($ARGS)"

        # read last run
        LASTRUN=`cat $LASTRUN_INDEX 2>/dev/null || echo 0`
        NOW=`date "+%Y%m%d%H" | cut -b 1-9`

        # Skip, if update already in progress!
        if (( `eval "$CHECKUPDRUNNING"` )); then
            log 3 "Skipped! (Update in progress!)"
        # Skip update if in same hour range
        elif (( LASTRUN == NOW )); then
            log 3 "Skipped! Already done a while ago." "[ Last $LASTRUN
== Now $NOW ]"
        # Check & install updates
        else
            # run this part in background, otherwise it will conflict
with other wakeup scripts!
            cat > $TMP_FILE <<__EOT__
#!/bin/bash
trap "" 1 2 3 15
```

```
LOG_FILE=$LOG_FILE
SCRIPTNAME=$SCRIPTNAME
exec >>$LOG_FILE
exec 2>&1
export DBUS_SESSION_BUS_ADDRESS=$DBUS_SESSION_BUS_ADDRESS
export DISPLAY="$DESKTOP_DISPLAY"
`set | egrep '^DISPLAY|^XAUTHORITY' | while read L; do echo "export
$L"; done`
`set | egrep '^DESKTOP_|^NOTIFY_'`
`tfunc log`
`tfunc quit`
            log 5 "Background process initiated from PID \$1" "[
DESKTOP_USER=$DESKTOP_USER; DISPLAY=$DISPLAY ]"

            # check internet connection first
            log 4 "Checking connectivity..."
            I=0
            while true
            do
                (( I > 20 )) && {
                    log 1 "ERROR: could not detect internet
connectivity!"
                    quit 1
                }

                let I=I+1

                DG=\$( ip r | fgrep default | awk '{ print \$3; }' )
                [ -z "\$DG" ] && {
                    log 5 "> No default gateway found, waiting 10sec
for interface..."
                    sleep 10
                    continue
                }

                ping -c 1 -q -W 3 \$DG || {
                    log 5 "> default gateway '\$DG' is not reachable,
waiting 10sec to try again..."
                    sleep 10
                    continue
                }

                ping -c 1 -q -W 3 www.google.com || {
                    log 5 "> www.google.com is not reachable, waiting
10sec to try again..."
                    sleep 10
                    continue
                }

                break
            done
            log 4 ">> Ok!"

            log 3 "Starting Mint Update!" "[ Last $LASTRUN != Now $NOW ]"
            mintupdate-tool -r -y $MUTOPTS upgrade \
                &&  log 4 "> mintupdate-tool finished with no error" \
                || {
                    log 1 "> mintupdate-tool returned error \$?!" ">> Check $LOG_FILE for details & retry using mintupdate gui!"
                    [ "$OPENUPDGUIONERR" == "true" ] \
                        && echo "su -c 'export DISPLAY=$DESKTOP_DISPLAY; mintupdate'" $DESKTOP_USER | at now
                   }
            echo >$LASTRUN_INDEX "$NOW"
            quit 0
__EOT__
            log 5 "Starting update background process in $UPDATEDELAY min..." "[ $TMP_FILE ]"
            chmod +x $TMP_FILE
            echo $TMP_FILE $$ | at -M now + $UPDATEDELAY min
            # Using at was the only way I could find that my child process was not killed
            # (nor disown & or hohup & did survive - dunno why!?) or caused strange
            # side effects. Using at everything works perfect! :)
        fi
        ;;

    pre|hibernate|suspend)
        log 4 "## Suspend ($ARGS)"
        # if going in hibernate/suspend just wait a moment if an update is running.
        # But even if, it should be continued after resume...
        # So no complicated hibernate handling is required.

        let I=0
        while (( `eval "$CHECKUPDRUNNING"` ))
        do
            (( I == 10 )) && break
            ((    ! I   )) && log 2 "> Running updates detected!" "Waiting max. 100sec to go to sleep ($ARGS)..."
            ((      I   )) && log 4 "> ... $I"
            sleep 10
            let I=I+1
        done
        (( ! I )) && log 5 "> No running updates detected."
        quit 0
        ;;

    cleanup)
        rm -f $LASTRUN_INDEX $LOG_FILE $TMP_FILE >/dev/null 2>&1
        log 4 ">> Cleanup finished <<"
        ;;

    *)  log 1 "Mode '$ARGS' ist not supported!" "Please report to
$AUTHOR if it was not a cmdline try by yourself!"
        quit 1
        ;;
esac
```