How To Get Plymouth To Display Boot Messages When Booting?

VitalyVitaly 84111 gold badge88 silver badges1515 bronze badges

OK so I've been working on this problem for 4 days straight now, and I almost nailed it completely. So far I was able to get Plymouth to boot with messages displayed, but unfortunately the messages are truncated. Right now am trying to tweak the scripts but i don't know where the problem is in the /lib/lsb/init-functions script or the /lib/plymouth /themes/"theme-name"/mdv.script.

Here is my work so far.

first you have to make init-functions send messages to Plymouth by making it look like this (go through each line to see the differences and copy the line which corresponds to Plymouth sending):

```
# /lib/lsb/init-functions for Debian -*- shell-script -*-
#
#Copyright (c) 2002-08 Chris Lawrence
#All rights reserved.
#
#Redistribution and use in source and binary forms, with or without
#modification, are permitted provided that the following conditions
#are met:
#1. Redistributions of source code must retain the above copyright
#   notice, this list of conditions and the following disclaimer.
#2. Redistributions in binary form must reproduce the above copyright
#   notice, this list of conditions and the following disclaimer in the
#   documentation and/or other materials provided with the distribution.
#3. Neither the name of the author nor the names of other contributors
#   may be used to endorse or promote products derived from this software
#   without specific prior written permission.
#
#THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
#IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
#WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
#ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE
#LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
#CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
#SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
#BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
#WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
#OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
#EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

start_daemon () {
    local force nice pidfile exec i args
    force=0
    nice=0
    pidfile=/dev/null

    OPTIND=1
    while getopts fn:p: opt ; do
        case "$opt" in
            f)  force=1;;
            n)  nice="$OPTARG";;
            p)  pidfile="$OPTARG";;
        esac
    done
```

```
    shift $(($OPTIND - 1))
    if [ "$1" = '--' ]; then
        shift
    fi

    exec="$1"; shift

    args="--start --nicelevel $nice --quiet --oknodo"
    if [ $force = 1 ]; then
        /sbin/start-stop-daemon $args --chdir "$PWD" --startas $exec --pidfile
    elif [ $pidfile ]; then
        /sbin/start-stop-daemon $args --chdir "$PWD" --exec $exec --oknodo --pi
    else
        /sbin/start-stop-daemon $args --chdir "$PWD" --exec $exec -- "$@"
    fi
}

pidofproc () {
    local pidfile line i pids= status specified pid
    pidfile=
    specified=

    OPTIND=1
    while getopts p: opt ; do
        case "$opt" in
            p)  pidfile="$OPTARG"; specified=1;;
        esac
    done
    shift $(($OPTIND - 1))

    base=${1##*/}
    if [ ! "$specified" ]; then
        pidfile="/var/run/$base.pid"
    fi

    if [ -n "${pidfile:-}" -a -r "$pidfile" ]; then
        read pid < "$pidfile"
        if [ -n "${pid:-}" ]; then
            if $(kill -0 "${pid:-}" 2> /dev/null); then
                echo "$pid"
                return 0
            elif ps "${pid:-}" >/dev/null 2>&1; then
                echo "$pid"
                return 0 # program is running, but not owned by this user
            else
                return 1 # program is dead and /var/run pid file exists
            fi
        fi
    fi
    if [ -x /bin/pidof -a ! "$specified" ]; then
        status="0"
        /bin/pidof -o %PPID -x $1 || status="$?"
        if [ "$status" = 1 ]; then
            return 3 # program is not running
        fi
        return 0
    fi
    return 4 # Unable to determine status
}

# start-stop-daemon uses the same algorithm as "pidofproc" above.
killproc () {
    local pidfile sig status base i name_param is_term_sig
    pidfile=
    name_param=
    is_term_sig=no

    OPTIND=1
    while getopts p: opt ; do
```

```
        case "$opt" in
            p)  pidfile="$OPTARG";;
        esac
    done
    shift $(($OPTIND - 1))

    base=${1##*/}
    if [ ! $pidfile ]; then
        name_param="--name $base --pidfile /var/run/$base.pid"
    else
        name_param="--pidfile $pidfile"
    fi

    sig=$(echo ${2:-} | sed -e 's/^-\(.*\)/\1/')
    sig=$(echo $sig | sed -e 's/^SIG\(.*\)/\1/')
    if [ -z "$sig" -o "$sig" = 15 -o "$sig" = TERM ]; then
        is_term_sig=yes
    fi
    status=0
    if [ ! "$is_term_sig" = yes ]; then
        if [ -n "$sig" ]; then
            /sbin/start-stop-daemon --stop --signal "$sig" --quiet $name_param
        else
            /sbin/start-stop-daemon --stop --quiet $name_param || status="$?"
        fi
    else
        /sbin/start-stop-daemon --stop --quiet --oknodo $name_param || status="
    fi
    if [ "$status" = 1 ]; then
        if [ -n "$sig" ]; then
            return 0
        fi
        return 3 # program is not running
    fi

    if [ "$status" = 0 -a "$is_term_sig" = yes -a "$pidfile" ]; then
        pidofproc -p "$pidfile" "$1" >/dev/null || rm -f "$pidfile"
    fi
    return 0
}

# Return LSB status
status_of_proc () {
    local pidfile daemon name status

    pidfile=
    OPTIND=1
    while getopts p: opt ; do
        case "$opt" in
            p)  pidfile="$OPTARG";;
        esac
    done
    shift $(($OPTIND - 1))

    if [ -n "$pidfile" ]; then
        pidfile="-p $pidfile"
    fi
    daemon="$1"
    name="$2"

    status="0"
    pidofproc $pidfile $daemon >/dev/null || status="$?"
    if [ "$status" = 0 ]; then
        log_success_msg "$name is running"
        return 0
    elif [ "$status" = 4 ]; then
        log_failure_msg "could not access PID file for $name"
        return $status
    else
        log_failure_msg "$name is not running"
```

```
        return $status
    fi
}

log_use_fancy_output () {
    TPUT=/usr/bin/tput
    EXPR=/usr/bin/expr
    if [ -t 1 ] && [ "x${TERM:-}" != "x" ] && [ "x${TERM:-}" != "xdumb" ] && [
        [ -z $FANCYTTY ] && FANCYTTY=1 || true
    else
        FANCYTTY=0
    fi
    case "$FANCYTTY" in
        1|Y|yes|true)   true;;
        *)              false;;
    esac
}

log_success_msg () {
    if [ -n "${1:-}" ]; then
        log_begin_msg $@
    fi
    log_end_msg 0
}

log_failure_msg () {
    if [ -n "${1:-}" ]; then
        log_begin_msg $@ "..."
    fi
    log_end_msg 1 || true
}

log_warning_msg () {
    if [ -n "${1:-}" ]; then
        log_begin_msg $@ "..."
    fi
    log_end_msg 255 || true
}

#
# NON-LSB HELPER FUNCTIONS
#
# int get_lsb_header_val (char *scriptpathname, char *key)
get_lsb_header_val () {
        if [ ! -f "$1" ] || [ -z "${2:-}" ]; then
                return 1
        fi
        LSB_S="### BEGIN INIT INFO"
        LSB_E="### END INIT INFO"
        sed -n "/$LSB_S/,/$LSB_E/ s/# $2: \(.*\)/\1/p" $1
}

# SEND MESSAGES TO PLYMOUTH
if [ -x /bin/plymouth ] && pidof plymouthd >/dev/null
then
    plymouth_send() {
        [ "$1" = '-n' ] && { # add a flag '>' for lines that will be extended
            shift
            /bin/plymouth message --text=">$*" || true
            return
        }
        [ "$1" = '-w' ] && { # add "warning" formatting
            shift
            /bin/plymouth update --status="warning" || true
            /bin/plymouth message --text="$*" || true
            /bin/plymouth update --status="normal" || true
            return
        }
        [ "$1" = '-f' ] && { # add "failed" formatting
            shift
```

```
            /bin/plymouth update --status="failed" || true
            /bin/plymouth message --text="$*" || true
            /bin/plymouth update --status="normal" || true
            return
        }
        /bin/plymouth message --text="$*" || true
    }
else
    plymouth_send() { :; }
fi

# int log_begin_message (char *message)
log_begin_msg () {
    if [ -z "${1:-}" ]; then
        return 1
    fi
    echo -n "$@"
    plymouth_send -n "$@"
}

# Sample usage:
# log_daemon_msg "Starting GNOME Login Manager" "gdm"
#
# On Debian, would output "Starting GNOME Login Manager: gdm"
# On Ubuntu, would output " * Starting GNOME Login Manager..."
#
# If the second argument is omitted, logging suitable for use with
# log_progress_msg() is used:
#
# log_daemon_msg "Starting remote filesystem services"
#
# On Debian, would output "Starting remote filesystem services:"
# On Ubuntu, would output " * Starting remote filesystem services..."

log_daemon_msg () {
    if [ -z "${1:-}" ]; then
        return 1
    fi
    log_daemon_msg_pre "$@"

    if [ -z "${2:-}" ]; then
        echo -n "$1:"
        plymouth_send -n "$1:"
        return
    fi

    echo -n "$1: $2"
    plymouth_send -n "$1: $2"
    log_daemon_msg_post "$@"
}

# #319739
#
# Per policy docs:
#
#     log_daemon_msg "Starting remote file system services"
#     log_progress_msg "nfsd"; start-stop-daemon --start --quiet nfsd
#     log_progress_msg "mountd"; start-stop-daemon --start --quiet mountd
#     log_progress_msg "ugidd"; start-stop-daemon --start --quiet ugidd
#     log_end_msg 0
#
# You could also do something fancy with log_end_msg here based on the
# return values of start-stop-daemon; this is left as an exercise for
# the reader...
#
# On Ubuntu, one would expect log_progress_msg to be a no-op.
log_progress_msg () {
    if [ -z "${1:-}" ]; then
        return 1
    fi
```

```
    echo -n " $@"
    plymouth_send -n " $@"
}


# int log_end_message (int exitstatus)
log_end_msg () {
    # If no arguments were passed, return
    if [ -z "${1:-}" ]; then
        return 1
    fi

    retval=$1

    log_end_msg_pre "$@"

    # Only do the fancy stuff if we have an appropriate terminal
    # and if /usr is already mounted
    if log_use_fancy_output; then
        RED=`$TPUT setaf 1`
        YELLOW=`$TPUT setaf 3`
        NORMAL=`$TPUT op`
    else
        RED=''
        YELLOW=''
        NORMAL=''
    fi

    if [ $1 -eq 0 ]; then
        echo "."
        plymouth_send "."
    elif [ $1 -eq 255 ]; then
        /bin/echo -e " ${YELLOW}(warning).${NORMAL}"
        plymouth_send -w "warning"
    else
        /bin/echo -e " ${RED}failed!${NORMAL}"
        plymouth_send -f "failed"
    fi
    log_end_msg_post "$@"
    return $retval
}

log_action_msg () {
    echo "$@."
    plymouth_send "$@."
}

log_action_begin_msg () {
    echo -n "$@..."
    plymouth_send -n "$@..."
}

log_action_cont_msg () {
    echo -n "$@..."
    plymouth_send -n "$@..."
}

log_action_end_msg () {
    log_action_end_msg_pre "$@"
    if [ -z "${2:-}" ]; then
        end="."
    else
        end=" ($2)."
    fi

    if [ $1 -eq 0 ]; then
        echo "done${end}"
        plymouth_send "done${end}"
    else
        if log_use_fancy_output; then
```

```
            RED=`$TPUT setaf 1`
            NORMAL=`$TPUT op`
            /bin/echo -e "${RED}failed${end}${NORMAL}"
            plymouth_send -f "failed${end}"
        else
            echo "failed${end}"
            plymouth_send -f "failed${end}"
        fi
    fi
    log_action_end_msg_post "$@"
}

# Hooks for /etc/lsb-base-logging.sh
log_daemon_msg_pre () { :; }
log_daemon_msg_post () { :; }
log_end_msg_pre () { :; }
log_end_msg_post () { :; }
log_action_end_msg_pre () { :; }
log_action_end_msg_post () { :; }

FANCYTTY=
[ -e /etc/lsb-base-logging.sh ] && . /etc/lsb-base-logging.sh || true
```

Now after you've added that to the init-functions you have to edit your Plymouth theme mdv.script

This is my latest updated version of the script:

```
# INT2MIL-Ubuntu-10.10-Eng splashy like theme

Window.GetMaxWidth = fun (){
  i = 0;
  width = 0;
  while (Window.GetWidth(i)){
    width = Math.Max(width, Window.GetWidth(i));
    i++;
    }
  return width;
};

Window.GetMaxHeight = fun (){
  i = 0;
  height = 0;
  while (Window.GetHeight(i)){
    height = Math.Max(height, Window.GetHeight(i));
    i++;
    }
  return height;
};
#change animcount to increase/decrease speed of spinning arrows
anim.imagecount = 100;
anim.target_width =  0.2* 0.46 * Window.GetWidth();
anim.target_height =  0.2* 0.46 * Window.GetWidth();

fun RotatedImage (index){
    index = Math.Int(index);
    if (!RotatedImageCache[index])
        RotatedImageCache[index] = anim.original_image.Rotate((Math.Pi*2*index)

    return RotatedImageCache[index];
    }


if (Plymouth.GetMode() == "suspend" || Plymouth.GetMode() == "resume") {
  background.original_image = ImageNew("suspend.png");
  Window.SetBackgroundTopColor(1, 0, 0);
  Window.SetBackgroundBottomColor(0, 1, 0);
}
```

```
else {
  logo.original_image = ImageNew("logo.png");
  background.original_image = ImageNew("background.png");
  Window.SetBackgroundTopColor(0.234, 0.43, 0.705);
  Window.SetBackgroundBottomColor(0.16, 0.25, 0.44);

  anim.image= ImageNew("animation.png");
  anim.original_image= anim.image.Scale(anim.target_width, anim.target_width);

  anim.sprite = SpriteNew();
  anim.sprite.SetImage(RotatedImage (0));
  anim.sprite.SetX((Window.GetX() + Window.GetWidth() - RotatedImage(0).GetWidt
  anim.sprite.SetY(Window.GetY() + Window.GetHeight() * 0.37);
  anim.angle = 0;
  anim.index = 0;


}
#change reduction size to make logo bigger
ratio =  logo.original_image.GetWidth() / logo.original_image.GetHeight();
reduction = 0.4;
logo.image = logo.original_image.Scale(reduction * Window.GetMaxWidth() , reduc
logo.sprite = SpriteNew();
logo.sprite.SetImage(logo.image);
logo.opacity_angle = 0;
#change logo location
logo.sprite.SetX((Window.GetX() + Window.GetMaxWidth()  - logo.image.GetWidth()
logo.sprite.SetY(Window.GetY() + Window.GetHeight() * 0.37);
#background image attributs x,z,y
background.image = background.original_image.Scale(Window.GetMaxWidth() , Windo
background.sprite = SpriteNew();
background.sprite.SetImage(background.image);
background.sprite.SetPosition(Window.GetX(), Window.GetY(), -100);

sprite_prompt = SpriteNew();

fun refresh_callback ()
  {
    if (status == "normal")
      {
#anim.index=speed of rotation
    anim.index += 1;
    anim.index %= anim.imagecount;
    anim.sprite.SetImage(RotatedImage (anim.index));
        #anim.sprite.SetOpacity (1);
    motif.sprite.SetOpacity(motif.opacity);
      }
    else
      {
        anim.sprite.SetOpacity(1);
    motif.sprite.SetOpacity(1);
      }

  }

if (Plymouth.GetMode() != "suspend" && Plymouth.GetMode() != "resume") {
  Plymouth.SetRefreshFunction (refresh_callback);
}

#----------------------------------------- Dialog ------------------------------

status = "normal";

fun dialog_setup()
  {
    local.box;
    local.lock;
    local.entry;
    local.prompt_sprite;
```

```
    box.image = ImageNew("box.png");
    lock.image = ImageNew("lock.png");
    entry.image = ImageNew("entry.png");

    box.sprite = SpriteNew();
    box.sprite.SetImage(box.image);
    box.x = Window.GetX() + Window.GetWidth()  / 2 - box.image.GetWidth()/2;
    box.y = Window.GetY() + Window.GetHeight() / 2 - box.image.GetHeight()/2;
    box.z = 10000;
    box.sprite.SetPosition(box.x, box.y, box.z);

    lock.sprite = SpriteNew();
    lock.sprite.SetImage(lock.image);
    lock.x = box.x + box.image.GetWidth()/2 - (lock.image.GetWidth() + entry.im
    lock.y = box.y + box.image.GetHeight()/2 - lock.image.GetHeight()/2;
    lock.z = box.z + 1;
    lock.sprite.SetPosition(lock.x, lock.y, lock.z);

    entry.sprite = SpriteNew();
    entry.sprite.SetImage(entry.image);
    entry.x = lock.x + lock.image.GetWidth();
    entry.y = box.y + box.image.GetHeight()/2 - entry.image.GetHeight()/2;
    entry.z = box.z + 1;
    entry.sprite.SetPosition(entry.x, entry.y, entry.z);

    prompt_sprite = SpriteNew();
    prompt_sprite.SetPosition(box.x, box.y - 20, box.z);

    global.dialog.box = box;
    global.dialog.lock = lock;
    global.dialog.entry = entry;
    global.dialog.bullet_image = ImageNew("bullet.png");
    global.dialog.prompt_sprite = prompt_sprite;
    dialog_opacity (1);
  }

fun dialog_opacity(opacity)
  {
    dialog.box.sprite.SetOpacity(opacity);
    dialog.lock.sprite.SetOpacity(opacity);
    dialog.entry.sprite.SetOpacity(opacity);
    dialog.prompt_sprite.SetOpacity(opacity);
    for (index = 0; dialog.bullet[index]; index++)
      {
        dialog.bullet[index].sprite.SetOpacity(opacity);
      }
  }

fun display_normal_callback ()
  {
    global.status = "normal";
    if (global.dialog)
      dialog_opacity (0);
  }

fun display_password_callback (prompt, bullets)
  {
    global.status = "password";
    if (!global.dialog)
    dialog_setup();
    else
    dialog_opacity(1);
    motif.sprite.SetOpacity(1);
    anim.sprite.SetOpacity(1);

    dialog.prompt_sprite.SetImage(Image.Text(prompt, 1.0, 1.0, 1.0));
    for (index = 0; dialog.bullet[index] || index < bullets; index++)
      {

        if (!dialog.bullet[index])
```

```
          {
            dialog.bullet[index].sprite = SpriteNew();
            dialog.bullet[index].sprite.SetImage(dialog.bullet_image);
            dialog.bullet[index].x = dialog.entry.x + index * dialog.bullet_ima
            dialog.bullet[index].y = dialog.entry.y + dialog.entry.image.GetHei
            dialog.bullet[index].z = dialog.entry.z + 1;
            dialog.bullet[index].sprite.SetPosition(dialog.bullet[index].x, dia
          }
        if (index < bullets)
          dialog.bullet[index].sprite.SetOpacity(1);
        else
          dialog.bullet[index].sprite.SetOpacity(0);
      }
  }

fun display_message_callback (prompt)
  {

prompt = Image.Text(prompt,1.0, 1.0, 1.0);
sprite_prompt.SetImage(prompt);
sprite_prompt.SetPosition(Window.GetX() + (Window.GetWidth() - prompt.GetWidth(
  }

/* instantiate dialog at startup, to ensure all icons are loaded in memory befo
dialog_setup(); dialog_opacity(0);
Plymouth.SetDisplayNormalFunction(display_normal_callback);
Plymouth.SetDisplayPasswordFunction(display_password_callback);
Plymouth.SetMessageFunction(display_message_callback);

#---------------------------------------- Progress Bar ----------------------

progress_box.image = Image("progress_box.png");
progress_box.sprite = Sprite(progress_box.image);

progress_box.x = Window.GetX() + Window.GetWidth() / 2 - progress_box.image.Get
progress_box.y = Window.GetY() + Window.GetHeight() * 0.65 - progress_box.image
progress_box.sprite.SetPosition(progress_box.x, progress_box.y, 0);

progress_bar.original_image = Image("progress_bar.png");
progress_bar.sprite = Sprite();

progress_bar.x = Window.GetX() + Window.GetWidth()  / 2 -        progress_bar.o
progress_bar.y = Window.GetY() + Window.GetHeight() * 0.65  - progress_box.imag
progress_bar.sprite.SetPosition(progress_bar.x, progress_bar.y, 1);

fun progress_callback (duration, progress)
  {
    if (progress_bar.image.GetWidth () != Math.Int (progress_bar.original_image
      {
# * 3 = multiply progress by 3
        progress_bar.image = progress_bar.original_image.Scale(progress_bar.ori
        progress_bar.sprite.SetImage (progress_bar.image);
      }
  }

Plymouth.SetBootProgressFunction(progress_callback);

#---------------------------------------- Status Update ----------------------

NUM_SCROLL_LINES=10;
LINE_WIDTH=55;

# width of one character doesnt work------------
CHAR_WIDTH = 7;
# height of one character
CHAR_HEIGHT = 10;
#-----------------------

msg_color = [1,1,1]; # msg_color is array
```

```
#status callback function

fun update_status_callback(sta) {
  if (sta == "failed") msg_color = [1,0,0];
  if (sta == "warning") msg_color = [0.8,0.8,0];
  if (sta == "normal") msg_color = [0.5,0.5,0.5];
}

screen_width = Window.GetWidth();
screen_height = Window.GetHeight();

#Initialising text images and their positions
# 20 is the height (including line spacing) of each line

for (i=0; i < NUM_SCROLL_LINES; i++) {
  lines[i]= Image.Text("", msg_color[0], msg_color[1], msg_color[2]);
  message_sprite[i] = Sprite();
  message_sprite[i].SetPosition(screen_width * 0.025, (screen_height * 0.6) + (
}

fun StringLength(string) {

  index = 0;
  str = String(string);
  while(str.CharAt(index)) index++;
  return index;
}

pretext = String("");

#scroll message function

fun scroll_message_callback(text) {

 ##nobreak function

  nobreak = 0;
   if (text.CharAt(0) == ">") {    # "no linebreak" flag, like "-n"
       text = text.SubString(1, StringLength(text)); # remove ">" at front
       nobreak = 1;
   }

   if ((pretext == "") || (StringLength(text) > 15))  {
   if (text == ".") return;    # ignore messages of only a single dot

       if (nobreak == 1) pretext = text;



#Truncate the message if too long


   if (StringLength(text) > LINE_WIDTH) {
     text = text.SubString(0, LINE_WIDTH - 0);
     text += "...";
   }




#Shift message one up

   for (i = 0; i < NUM_SCROLL_LINES - 1; i++) {
     lines[i] = lines[i+1];
   }


}
   else {    # the previous message was flagged to have no linebreak
```

```
          // Truncate the message if too long
       if (StringLength(text) > LINE_WIDTH - 5) { # leave min. 5 for pretext
         text = text.SubString(0, LINE_WIDTH - 8);
         text += "...";
       }

          # Truncate the previous message if too long
       if (StringLength(pretext) > (LINE_WIDTH - StringLength(text))) {
         pretext = pretext.SubString(0, LINE_WIDTH - StringLength(text) - 3);
         pretext += "...";
       }

       text = pretext + text;

       if (nobreak == 1) pretext = text;
       else pretext = ">";

   }


#Create the image for the latest message

     lines[i] = Image.Text(text, msg_color[0], msg_color[1], msg_color[2]);

#Re-positioning the text images

  for (i = 0; i < NUM_SCROLL_LINES; i++) {
     message_sprite[i].SetImage(lines[i]);
   }
}

 Plymouth.SetUpdateStatusFunction(update_status_callback);

 Plymouth.SetUpdateStatusFunction(scroll_message_callback);


#------------------------------------- Quit ------------------------------

fun quit_callback ()
{
 anim.sprite.SetOpacity (1);
  if (Plymouth.GetMode() == "shutdown") {
   motif.sprite.SetOpacity(1);
  }
}

Plymouth.SetQuitFunction(quit_callback);
```

Basically the script can be applied to any theme, all you have to do is provide the filenames of the images in the folder. And changing a few other lines to adjust the images on the screen. Or what you do is you copy the necessary part like the lets say you want the progress part so all you have to do is copy everything from

---------- Progress Bar ----------

till

---------- Status Update ----------

After you are done with editing the mdv.script be sure to `sudo update-initramfs -u` and on your next boot you shall see your new splash.

Be sure to check out the links provided in my question they are very informative and will get you to understand plymouth scripting in no time.

Now if you have done everything i said here you boot splash should display scrolling messages. Now about the truncating part, I am currently working on it, but its kinda annoying to have to reboot my machine everytime i make some change. ***Is it possible to test a boot process while am logged in*** like

```
sudo plymouthd ; sudo plymouth --show-splash ; sudo plymouth update --status="H
```

Another way you can test Plymouth is by copying the above command into a text file and adding `sudo plymouth update --status="your message"` to have more messages scroll through. Then make the file executable and run in terminal.

Hope this helps anyone else wanting to edit their Plymouth splash. Good Luck!!!