```
#!/bin/sh
#
# mkulawmsg/speak   (C) 2000 Malte J. Wetz <mail@malte-wetz.de>
#
# If called as "mkulawmsg", this script will generate an
# ulaw-encoded audio stream from a plain text file. This stream can
# be used with vbox3.
#
# If called as "speak", it will pretty much do the same. But instead
# of recoding it for vbox, it will play the audio using aplay from
# ALSA.
#
# It requires "mbrola", "txt2pho" and "sox" to be installed and
# configured correctly on your system.
# You can download mbrola from here:
#      http://mambo.ucsc.edu/psl/mbrola/
# There is also a link to txt2pho.
#
#
# NOTE: I discovered that mbrola appearently does not work together
#       with newer glibc (ver 2.3 and higher). As a workaround, I am
#       using the uClibc from http://www.uclibc.org. However, this
#       causes problems with pipes so that
#       'cat test.pho | mbrola - out.wav' does not work.
#       As a workaround for that, I set up a cache that buffers the
#       pho and audio as normal files. That works.
#

##########################################################################
##
# SETTINGS: Change the following parameters to meet your requirements.
# If you don't know what you're doing, just don't do anything or at
# least keep a backup!
##########################################################################
##

# Full path of mbrola binary
MBROLA=/opt/bin/mbrola

# Full path to mbrola voices
VOICEPATH=/opt/mbrola

# mbrola default voice to use with this script (without path!)
# Can be changed on command line via "-v".
VOICE=de5

# Full path to txt2pho binary
TXT2PHO=/opt/bin/txt2pho

# Full path of preproc binary (which is part of txt2pho)
PREPROC=/opt/bin/preproc
```

```
# Parameters for preproc. Specify Rules and abbreviations files
# (both are part of txt2pho).
PPPARAMS="/opt/mbrola/preproc/Rules.lst \
/opt/mbrola/preproc/Hadifix.abk"

# Full path of numfilt binary (which is part of txt2pho)
# NUMFILT=/usr/local/bin/numfilt

# Full path of pipefilt binary (which is part of txt2pho)
PIPEFILT=/opt/bin/pipefilt

# Full path of sox binary
SOX=/usr/bin/sox

# Parameters for sox. Please note that the input sampling rate
# ("%SR") will be replaced by the script. This is because different
# voices output different sample rates.
# Also note, that if $BASENAME is "mkulawmsg", output will be
# vbox-compatible and be written to stdout. If $BASENAME is "speak",
# output will be written to /dev/audio.
SOXPARAMS="-t raw -sw -r %SR - -t ul -r 8000 -"

# Here, you can specify which voice produces what output sampling
# rate (see above).
# Format is: "<voice>:<rate> [...]".
RATES="de1:16000 de2:16000 de3:22050 de4:16000 de5:22050 de6:22050
de7:22050"

# Set a default input sampling rate that will be applied if none
# of the entries above match.
DEFAULTRATE="111025"

# Give your path to aplay here
#PLAY="/usr/bin/esddsp /usr/bin/aplay"
PLAY=/usr/bin/aplay
#PLAY=/usr/bin/play

# Specify what is neccessary to play the resulting audio stream
# (see SOXPARAMS above for details).
# currently only for aplay 0.9.x
PLAYPARAMS="-t raw -f S16_LE -r%SR -q"          # use this for 'aplay'
#PLAYPARAMS="-t raw -s w -r %SR -f s -"         # use this for 'play'

# Where should the lockfile be placed?
LOCKFILE=/tmp/speak.lck

# Where to cache the files?  In the cache directory, speak will
# store .pho and .raw files and use no pipes (see NOTE above).
# The directory will be created if it does not exist and removed
# when not needed anymore. Leave blank for no cache.
CACHE=/tmp/speak.$$.cache
```

```
##############################################################################
##
# END OF SETTINGS: There should be nothing to change below this point!
##############################################################################
##

# Never change those two!
BASENAME=`basename $0`
VERSION="2.3.7"

# Evaluate command line options
while getopts hapnv:l: switch; do
  case "$switch" in
    a) cmd="$cmd | $PREPROC $PPPARAMS"  ;;
    p) cmd="$cmd | $PIPEFILT"           ;;
    n) cmd="$cmd | $NUMFILT"            ;;
    v) VOICE=$OPTARG                    ;;
    l) LOCKING=$OPTARG                  ;;
    h) cat<<EOF
$BASENAME $VERSION  (C) Malte J. Wetz <mail@malte-wetz.de>

DESCRIPTION
  If called as "mkulawmsg":
    Script that will use a 3rd party TTS system to generate an ulaw
    encoded voice stream from plain text. Text is taken from stdin.
    Resulting audio is compatible with vbox3 and written to stdout.
  If called as "speak":
    Script will work as above, but instead of converting the resulting
    voice stream to vbox3-Format, it will use a customizable audio
    player to send it directly to the sound card.

USAGE
  $BASENAME [OPTIONS]

PARAMETERS
  -h      Display this screen and exit
  -p      Will remove any newline characters from pipe and add breathe
          pause at punctuation marks (use pipefilt).
  -n      Speak digits as full numbers (use numfilt).
  -a      Replace some default abbreviations with full text
          (use preproc).
  -v xxx  Use Mbrola voice xxx for synthesis (default: $VOICE).
  -l n    If another instance of '$BASENAME -l' is running, wait max. n sec.
          for it to finish, then exit. Prevents more than one $BASENAME
talking
          at the same time.

NOTES
  - Options are order sensitve. I.e., if you specify "-n -a" then first
numbers
    will be filtered, then abbreviation. $BASENAME therefore will not say
```

```
    dates correctly. Instead, specify "-a -n".
EOF
    exit
    ;;
  esac
done

# Check for txt2pho config files
[ -r "/etc/txt2pho" -o -r "$HOME/.txt2phorc" ] || {
  echo "Error. Neither /etc/txt2pho nor ~/.txt2phorc ($USER) found."
  echo "Synthesis will fail. Exit now."
  exit 1
}

# Replace the input sample rate based on selected voice.
for i in $RATES; do
  v=`echo $i | cut -d ':' -f 1`
  r=`echo $i | cut -d ':' -f 2`
  [ "$v" = "$VOICE" ] && SR="$r"
done
[ -z "$SR" ] && SR=$DEFAULTRATE

# Based on name of script, decide what postprocessor to use
case "$BASENAME" in
  mkulawmsg)
    PP=$SOX
    PP_PARAMS=`echo $SOXPARAMS | sed -e "s/%SR/$SR/g"`
    unset CACHE
  ;;
  speak)
    PP=$PLAY
    PP_PARAMS=`echo $PLAYPARAMS | sed -e "s/%SR/$SR/g"`
  ;;
esac

# check for lock file...
[ -z "$LOCKING" ] || {
  # check if lockfile is stale...
  [ -e "$LOCKFILE" ] && {
    ps -p `cat "$LOCKFILE"` > /dev/null 2>&1 || {
      echo "Warning: stale lockfile \"$LOCKFILE\" found, removed"
      rm -f "$LOCKFILE"
    }
  }
  # wait for lockfile to be removed by other instance...
  while [ -e "$LOCKFILE" ]; do
    sleep 1
    count=$(($count + 1))
    [ "$count" -ge "$LOCKING" ] && {
      echo "Error: Waited $LOCKING seconds for other instance to finish, but
lockfile is still"
```

```
      echo "present. Giving up...."
      exit 1
    }
  done
}

[ -z "$LOCKING" ] || echo $$ > $LOCKFILE


# use the cache, is set
if [ -z "$CACHE" ]; then
  # Assemble the final command without cache and execute it
  cmd="cat $cmd | $TXT2PHO | \
    $MBROLA ${VOICEPATH}/${VOICE}/${VOICE} - -.raw | $PP $PP_PARAMS"
  eval $cmd
else
  # create cache dir if neccessary
  [ -d "$CACHE" ] || mkdir -p "$CACHE"
  # Assemble the final command with cache and execute it
  cmd="cat $cmd | $TXT2PHO > $CACHE/out.pho; \
    $MBROLA ${VOICEPATH}/${VOICE}/${VOICE} $CACHE/out.pho $CACHE/out.raw; \
    $PP $PP_PARAMS $CACHE/out.raw"
  eval $cmd
  # remove cache
  rm -rf "$CACHE"
fi

[ -z "$LOCKING" ] || rm -f "$LOCKFILE"

exit
```