wsprdaemon.sh   Help text   Rob Robinett AI6VN March 2020

This program reads the configuration file wsprdaemon.conf which defines a schedule to capture and post WSPR signals from one or more KiwiSDRs
and/or AUDIO inputs and/or RTL-SDRs.
Each KiwiSDR can be configured to run 8 separate bands, so 2 Kiwis can spot every 2 minute cycle from all 14 LF/MF/HF bands.
In addition, the operator can configure 'MERG_..' receivers which posts decodes from 2 or more 'real' receivers
but selects only the best SNR for each received callsign (i.e no double-posting)

Each 2 minute WSPR cycle this script creates a separate .wav recording file on this host from the audio output of each configured [receiver,band]
At the end of each cycle, each of those files is processed by the 'wsprd' WSPR decode application included in the WSJT-x application
which must be installed on this server. The decodes output by 'wsprd' are then spotted to the WSPRnet.org database.
The script allows individual [receiver,band] control as well as automatic scheduled band control via a watchdog process
which is automatically started during the server's bootup process.

```
-h                      => print this help message (execute '-vh' to get a description of the architecture of this program)

-a                      => stArt watchdog daemon which will start all scheduled jobs ( -w a )
-z                      => stop watchdog daemon and all jobs it is currently running (-w z )   (i.e.zzzz => go to sleep)
-s                      => show Status of watchdog and jobs it is currently running  (-w s ; -j s )
-p HOURS                      => generate ~/wsprdaemon/signal-levels.jpg for the last HOURS of SNR data

These flags are mostly intended for advanced configuration:

-i                      => list audio and RTL-SDR devices attached to this computer
-j ......               => Start, Stop and Monitor one or more WSPR jobs.  Each job is composed of one capture daemon and one decode/posting daemon
-j a,RECEIVER_NAME[,WSPR_BAND]    => stArt WSPR jobs(s).          RECEIVER_NAME = 'all' (default) ==  All RECEIVER,BAND jobs defined in wsprdaemon.conf
                                               OR       RECEIVER_NAME from list below
                                                  AND WSPR_BAND from list below
-j z,RECEIVER_NAME[,WSPR_BAND]    => Stop (i.e zzzzz)  WSPR job(s). RECEIVER_NAME defaults to 'all'
-j s,RECEIVER_NAME[,WSPR_BAND]    => Show Status of WSPR job(s).
-j l,RECEIVER_NAME[,WSPR_BAND]    => Watch end of the decode/posting.log file.  RECEIVER_ANME = 'all' is not valid
-j o                    => Search for zombie jobs (i.e. not in current scheduled jobs list) and kill them

-w ......               => Start, Stop and Monitor the Watchdog daemon
-w a                    => stArt the watchdog daemon
-w z                    => Stop (i.e put to sleep == zzzzz) the watchdog daemon
-w s                    => Show Status of watchdog daemon
-w l                    => Watch end of watchdog.log file by executing 'less +F watchdog.log'
```

```
-v                     => Increase verbosity of diagnotic printouts
-d                     => Signal all running processes as found in the *.pid files in the current directory to increment the logging verbosity
                        This permits changes to logging verbosity without restarting WD
-D                     => Signal all to decrement verbosity

Examples:
 ${0##*/} -a                   => stArt the watchdog daemon which will in turn run '-j a,all' starting WSPR jobs defined in '${WSPRDAEMON_CONFIG_FILE}'
 ${0##*/} -z                   => Stop the watchdog daemon but WSPR jobs will continue to run
 ${0##*/} -s                   => Show the status of the watchdog and all of the currently running jobs it has created
 ${0##*/} -j a,RECEIVER_LF_MF_0,2200   => on RECEIVER_LF_MF_0 start a WSPR job on 2200M
 ${0##*/} -j a                  => start WSPR jobs on all receivers/bands configured in ${WSPRDAEMON_CONFIG_FILE}
 ${0##*/} -j z                  => stop all WSPR jobs on all receivers/bands configured in ${WSPRDAEMON_CONFIG_FILE}, but note
                          that the watchdog will restart them if it is running
```

Valid RECEIVER_NAMEs which have been defined in '${WSPRDAEMON_CONFIG_FILE}':
$(list_known_receivers)

WSPR_BAND => {2200|630|160|80|80eu|60|60eu|40|30|20|17|15|12|10|6|2|1|0}

Author Rob Robinett AI6VN rob@robinett.us   with much help from John Seamons and a group of beta testers
I would appreciate reports which compare the number of reports and the SNR values reported by wsprdaemon.sh
    against values reported by the same Kiwi's autowspr and/or that same Kiwi fed to WSJT-x
In my testing wsprdaemon.sh always reports the same or more signals and the same SNR for those detected by autowspr,
    but I cannot yet guarantee that wsprdaemon.sh is always better than those other reporting methods.
"
[[${verbosity} -ge 1]] && echo "

An overview of the SW architecture of wsprdaemon.sh:

This program creates a error-resilient stand-alone WSPR receiving appliance which should run 24/7/365 without user attention and will recover from
loss of power and/or Internet connectivity.
It has been primarily developed and deployed on Rasberry Pi 3Bs which can support 20 or more WSPR decoding bands when KiwiSDRs are used as the demodulated signal sources.
However it is runing on other Debian 16.4 servers like the odroid and x86 servers (I think) without and modifications. Even Windows runs bash today, so perhaps
it could be ported to run there too. It has run on Max OSX, but I haven't check its operation there in many months.
It is almost entirely a bash script which excutes the 'wsprd' binary supplied in the WSJT-x distribution. To use a KiwiSDR as the signal soure it
uses a Python script supplied by the KiwiSDR author