

A bunch of snack tools for batch processing audio files

Jim Magnuson
james.magnuson@uconn.edu

Abstract

I describe 10 simple tools for batch processing of speech files using the snack tcl/tk from KTH. These are not polished. They have worked for our purposes, and are provided with the hope that they might be useful for others. If you use them and/or improve them, please let me know.

*Many, many thanks to **Kåre Sjölander** of KTH for developing snack and making it freely available.*

Please direct comments and questions to Jim Magnuson

A bunch of snack tools for batch processing audio files

I describe 10 simple tools for batch processing of speech files using the snack tcl/tk from KTH. These are not polished. They have worked for our purposes, and are provided with the hope that they might be useful for others. If you use them and/or improve them, please let me know.

To use the tools, you must have tcl and tk installed, and then you must install the snack toolkit. For details on obtaining and installing all of these, start with <http://www.speech.kth.se/snack/>. Please don't ask me for help installing tcl/tk or snack (well, unless you are very desperate).

Again, a million thanks to **Kåre Sjölander** of KTH for developing snack and making it freely available.

Here are brief descriptions of the tools. Details follow. The tools can be downloaded via links at <http://magnuson.psy.uconn.edu/reports.html> and/or <http://magnuson.psy.uconn.edu/snacktools>.

Tool	Name	Purpose
1	snack_info.tcl	Foreach file, print basic characteristics (duration, sampling rate, bit rate, type, etc.)
2	snack_check_amp.tcl	Foreach file, print max amplification possible without clipping
3	snack_amp.tcl	Multiply amplitude by a constant
4	snack_find_zeroes.tcl	Find zeroes and zero crossings in specified range of a file
5	snack_cut.tcl	Make a copy of a file with a specified section removed
6	snack_copy.tcl	Copy a specified section of a file to a new file
7	snack_concat_wavs.tcl	Concatenate file1 and file2 into file3
8	snack_add_zeroes.tcl	Pad the end of a file with specified number of silent samples
9	snack_resample.tcl	Change sampling rate
10	snack_viewer.tcl	GUI showing waveform, spectrogram, and including basic tools for playing files and labeling them

Tool 1: snack_info.tcl

This tool provides basic information about each file specified. It gives the duration in msec, the size in bytes, the sampling rate in hz, the maximum and minimum sample values, the bit rate, and the file format.

Usage: `snack_info.tcl FILELIST`

Usage examples:

```
snack_info.tcl ball.wav tile.wav done.wav
snack_info.tcl *.wav
```

Output example:

```
> snack_info.tcl *.wav
hIkt.wav          563.741496599    ms          24861 44100 23683 -30989 Lin16 1 WAV 44
hant.wav          605.782312925    ms          26715 44100 18452 -22169 Lin16 1 WAV 44
hep.wav           568.140589569    ms          25055 44100 23887 -30499 Lin16 1 WAV 44
hoist.wav         657.664399093    ms          29003 44100 17459 -21849 Lin16 1 WAV 44
hurry.wav         544.648526077    ms          24019 44100 12363 -17249 Lin16 1 WAV 44
hyuT.wav          628.798185941    ms          27730 44100 12890 -20521 Lin16 1 WAV 44
hyub.wav          768.321995465    ms          33883 44100 14183 -14635 Lin16 1 WAV 44
```

Tool 2: snack_check_amp.tcl

This tool checks each file and determines the greatest factor by which the samples could be multiplied without clipping (but this does not protect against other forms of distortion). For each file, it reports the maximum absolute value found in the file, and maximum multiplicand.

Usage: `snack_check_amp.tcl FILELIST`

Usage examples:

```
snack_check_amp.tcl ball.wav hep.wav
snack_check_amp.tcl *.wav
```

Output example:

```
> snack_check_amp.tcl *.wav
hIkt.wav 30989 1.05731065862
hant.wav 22169 1.47796472552
hep.wav 30499 1.07429751795
hoist.wav 21849 1.49961096618
hurry.wav 17249 1.89953040756
hyuT.wav 20521 1.59665708299
hyub.wav 14635 2.23881106935
```

Tool 3: snack_amp.tcl

This tool allows you to adjust the amplitude of the file by multiplying every sample by a constant greater than 0. Use values greater than 1 with caution, as they induce distortion. Note that if you were interested in boosting a file to its maximum without clipping, you can use the maximum multiplicand (multiplication factor) reported by `snack_check_amp.tcl`.

Usage: `snack_amp.tcl SOUNDFILE OUTFILE FACTOR`

Where `SOUNDFILE` is the original file, `OUTFILE` is the new amplitude-adjusted file, and `FACTOR` is the constant that amplitude is adjusted by.

Usage examples:

```
snack_amp.tcl ball.wav ball2.wav .75
```

Output example: (with followup stats checked with `snack_info.tcl`)

```
> snack_amp.tcl hep.wav hep_half.wav .5
Changing amplitude of hep.wav, writing to hep_half.wav

> snack_info.tcl hep*wav
```

snack tools		Cognitive Neuroscience of Language Lab (MagLab at UConn)										4
hep.wav	568.140589569	ms	25055	44100	23887	-30499	Lin16	1	WAV	44		
hep_half.wav	568.140589569	ms	25055	44100	11944	-15250	Lin16	1	WAV	44		

Tool 4: snack_find_zeroes.tcl

It is often useful to know where zero crossings are for trimming files. (Praat, among other tools, has great facilities for adjusting selections based on zero crossings.) This tool finds zero crossings in a specified range of a file.

Usage: `snack_find_zeroes.tcl SOUNDFILE START END [UNITS]`

Where `SOUNDFILE` is the original file, `START` is a number indicating where in the file you wish to start locating zero crossings, `END` is a number indicating the last position in the file you wish to check, and `UNITS` specifies whether `START` and `END` are in "msecs" or "bytes" (default, if units are unspecified, is " bytes "). For each file, it gives you sample (byte) numbers, msecs, sample values, and previous sample value (in parentheses) whenever there is a zero crossing (lines starting `#CROSS`) or an observed sample value of 0 (lines starting `#ZERO`).

Usage and output examples:

```
> snack_find_zeroes.tcl hant.wav 95 97 msecs
```

```
#CROSS 4191 95.034 -21 (142)
#CROSS 4195 95.125 54 (-80)
#CROSS 4204 95.329 -5 (140)
#ZERO 4208 95.420 0
#CROSS 4219 95.669 -18 (5)
#CROSS 4222 95.737 21 (-23)
#CROSS 4230 95.918 -5 (50)
#CROSS 4233 95.986 2 (-59)
#CROSS 4237 96.077 -5 (23)
#CROSS 4262 96.644 54 (-56)
#CROSS 4265 96.712 -86 (11)
#CROSS 4271 96.848 63 (-42)
```

```
# THIS EXAMPLE ASSUMES UNITS ARE BYTES SINCE NONE ARE SPECIFIED
```

```
> snack_find_zeroes.tcl hant.wav 4200 4230
```

```
#CROSS 4204 95.329 -5 (140)
#ZERO 4208 95.420 0
#CROSS 4219 95.669 -18 (5)
#CROSS 4222 95.737 21 (-23)
```

Tool 5: snack_cut.tcl

This tool truncates a specified range of samples from a file. The range can be specified in msecs or bytes (the default).

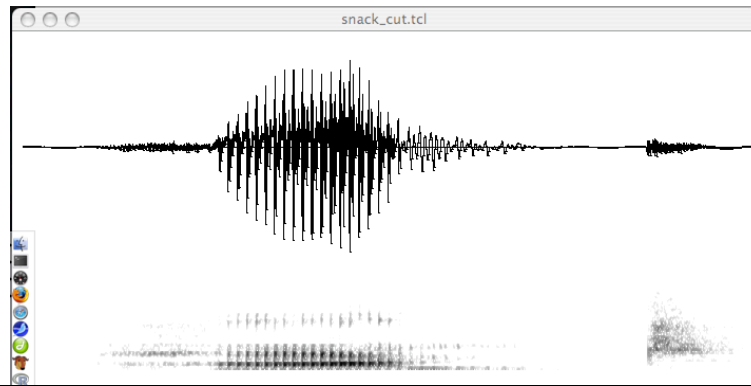
Usage: `snack_cut.tcl SOUNDFILE OUTFILE START END [UNITS]`

Where `SOUNDFILE` is the input file, `OUTFILE` is a NEW file with the specified samples removed, `START` is where to start cutting, `END` is where to stop (specify -1 to specify the end of the file), and `UNITS` is msecs or bytes (if you don't specify, it assumes bytes, that is, sample numbers). It also throws up a waveform and spectrogram of the new (cut) file (see example below). A fairly useless feature, but easy enough to delete if you like.

Usage and output examples: (with snack_info.tcl checking)

```
> snack_cut.tcl hant.wav hantcut.wav 4208 -1
```

```
Cutting from 4208 to 26714
```



Tool 6: snack_copy.tcl

This tool takes a specified range of samples from a file and copies them into another file. The syntax is identical to that of `snack_cut.tcl`. Range can be specified in msec or bytes (the default).

Usage: `snack_copy.tcl SOUNDFILE OUTFILE START END [UNITS]`

Where `SOUNDFILE` is the input file, `OUTFILE` is a NEW file consisting of the specified samples, `START` is where to start copying, `END` is where to stop (specify -1 to specify the end of the file), and `UNITS` is msec or bytes (if you don't specify, it assumes bytes, that is, sample numbers). It also throws up a waveform and spectrogram of the new (cut) file. A fairly useless feature, but easy enough to delete if you like.

Usage and output examples: (with `snack_info.tcl` checking)

```
snack_copy.tcl hant.wav hantpiece.wav 4208 -1
**** * SOUND hant.wav CHECK FROM 4208 TO 26715 ( bytes ).
length: 1 26715, r 44100, pcons 18, pixpsec 1484.16666667
# # # cropping from 4208 to 26715

> snack_info.tcl hant.wav hantpiece.wav
hant.wav          605.782312925    ms          26715 44100 18452 -22169 Lin16 1 WAV 44
hantpiece.wav     510.362811791    ms          22507 44100 18452 -22169 Lin16 1 WAV 44
```

Tool 7: snack_concat_wavs.tcl

This tool does just what it says, though it's limited to 2 files.

Usage: `snack_concat_wavs.tcl FILE1 FILE2 OUTFILE`

This is also one of the few tools in this set that actually gives you usage info if you don't specify arguments:

```
> snack_concat_wavs.tcl
# USAGE: prog file1 file2 outfile
Concatenates file1 and file2 (in that order
to new file, outfile. WARNING: there is
no error checking. This'll barf if you
specify files of different types or rates.
```

Usage and output examples: (with `snack_info.tcl` checking)

```
> snack_info.tcl hant.wav hep.wav
hant.wav          605.782312925    ms          26715 44100 18452 -22169 Lin16 1 WAV 44
hep.wav           568.140589569    ms          25055 44100 23887 -30499 Lin16 1 WAV 44

> snack_concat_wavs.tcl hant.wav hep.wav hanthep.wav
snack_info.tcl hanthep.wav
# Concatenating hant.wav and hep.wav to hanthep.wav

> snack_info.tcl hanthep.wav
hanthep.wav       1173.92290249    ms          51770 44100 23887 -30499 Lin16 1 WAV 44
```

Tool 8: snack_add_zeroes.tcl

Pad end of file with zeroes. *Should be revised to allow padding from any offset. Of course, you could use this with `snack_concat_wavs.tcl` to pad from beginnings of files.*

Usage: `snack_add_zeroes.tcl` SOUNDFILE OUTFILE NUMBER_OF_SAMPLES

Usage and output examples: (with `snack_info.tcl` checking)

```
> snack_add_zeroes.tcl hep.wav hep_padded.wav 100
```

```
> snack_info.tcl hep.wav hep_padded.wav
```

```
hep.wav          568.140589569    ms      25055 44100 23887 -30499 Lin16 1 WAV 44
```

```
hep_padded.wav   570.408163265    ms      25155 44100 23887 -30499 Lin16 1 WAV 44
```

Tool 9: snack_resample.tcl

Make a copy of a file with a new sampling rate.

Usage: `snack_resample.tcl SOUNDFILE OUTFILE NEWRATE`

This is also one of the few tools in this set that actually gives you usage info if you don't specify arguments:

```
> snack_concat_wavs.tcl
# USAGE: prog file1 file2 outfile
Concatenates file1 and file2 (in that order
to new file, outfile. WARNING: there is
no error checking. This'll barf if you
specify files of different types or rates.
```

Usage and output examples: (with `snack_info.tcl` checking)

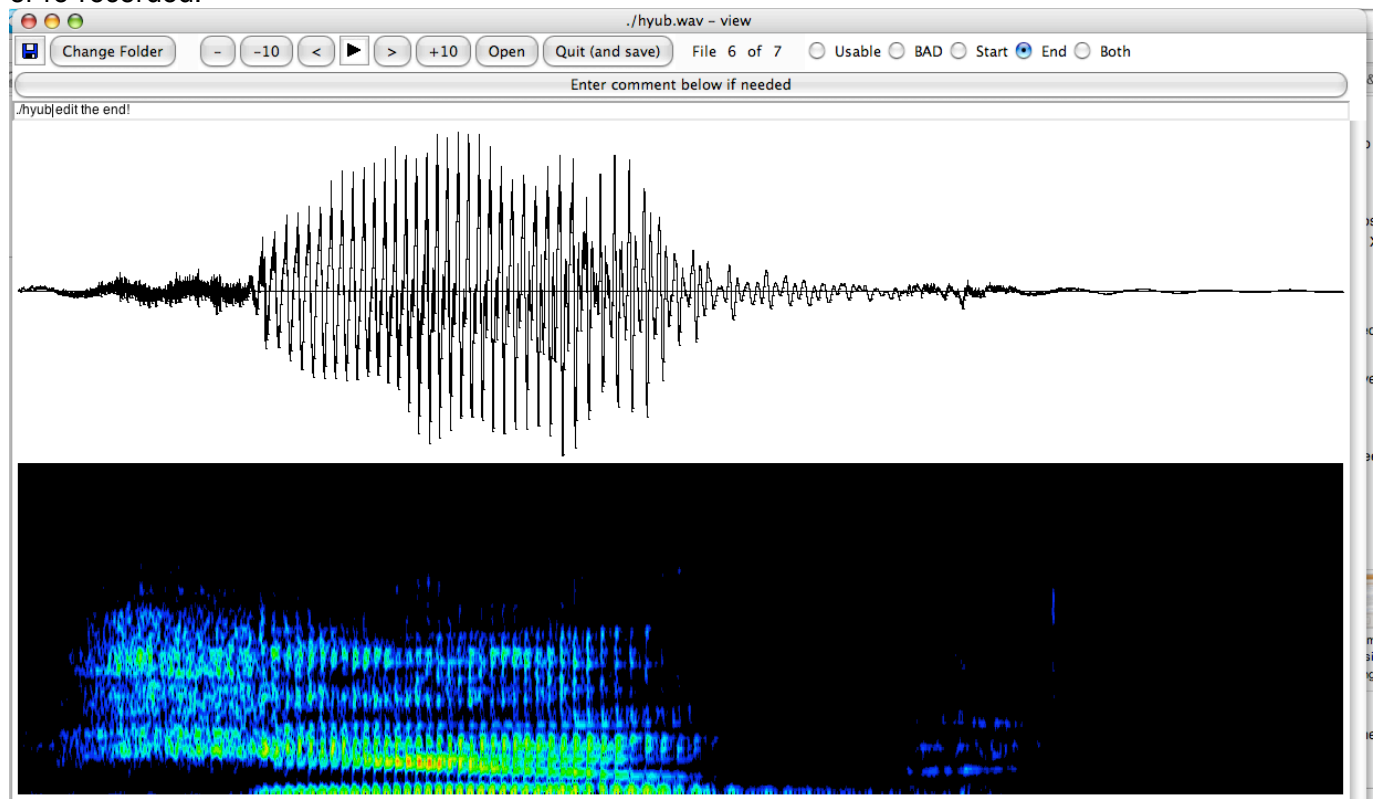
```
> snack_resample.tcl hep.wav hep11.wav 11025
```

```
> snack_info.tcl hep.wav hep11.wav
```

```
hep.wav          568.140589569    ms      25055 44100 23887 -30499 Lin16 1 WAV 44
hep11.wav        568.072562358    ms      6263 11025 10724 -14815 Lin16 1 WAV 44
```

Tool 10: snack_viewer.tcl

This tool provides a very rudimentary GUI for viewing and annotating a set of files. Only the most basic aspects of this tool are covered here. When you execute `snack_viewer.tcl`, it opens all wav files in the current directory. You can page through them using `>` and `<` icons, you can play each file, and you can add a label to a savable text file with the radio buttons at the right of the screen. You can also insert text comments. The default buttons were setup for evaluating a very large set of files to mark which ones needed trimming at the beginning and/or end of the file, and which were "BAD" and should be excluded or re-recorded.



Usage: `snack_viewer.tcl`

Output file example:

```
./hoist.wav      ./hoist          usable
./hurry.wav      ./hurry           usable
./hyuT.wav       ./hyuT            bad
./hyub.wav       ./hyub edit the end! end
```