The *sound* package

Managing sound I/O

The *sound* package is based on the `libsndfile` library written by

- **E. de Castro Lopo**
  see `http://www.zip.com.au/~erikd/libsndfile/`

The adaptation to LastWave was co-written by

- **E.Bacry** *CMAP, Ecole Polytechnique, 91128 Palaiseau Cedex France.*
- **R. Gribonval**, *IRISA-INRIA, Campus de Beaulieu, 35042 Rennes Cedex, France*
  email : remi.gribonval@inria.fr
  web : http://www.irisa.fr/metiss/gribonval

Contents

Part I

Documentation

Chapter 1

1.xxx documentation

Sorry, This is not written yet!

Chapter 2

New features (2.0) of the sound package

The syntax of the `sound` package of LastWave has changed a bit to allow easier use of stereo sounds. For example you can play a sinusoid at 500Hz sampled at 44 kHz and of duration 1sec as a mono sound by typing

```
(&wtrans) a> 0=[sin 44100 500]
(&wtrans) a> 0.dx=1/44100
(&wtrans) a> sound play 0
```

If you want to play the previous sinusoid on the left channel of a stereo sound and a click at the 34700th sample on the right channel, you should type

```
(&wtrans) a> 1=[dirac 44100 34700]
(&wtrans) a> 1.dx=1/44100
(&wtrans) a> sound play 0 1
```

HINT: If your audio system is configured correctly you should hear the sinusoid on the left channel, and the click on the right channel ! If it does not happen so, you should check whether your headphone/loudspeakers are cabled correctly ...

If you started playing a sound and you changed your mind, you can stop playing it with the `sound stop` command (only on Win32 systems so far!).

Manu: à corriger si ca marche sur Mac
A vereifier

If you want to record a sound, you should use the `sound record` command. Its syntax is different on Macintosh and Windows system, and it is not implemented on Unix systems (except Cygwin). On Unix systems, you should record the sound to a file using your favorite application, and use `sound read`.

2.1 Recording on Windows systems

Suppose you want to record 1.5 second of stereo sound at CD quality and store it in `&signal` variables. You should type

```
(&wtrans) a> sigLeft =[new &signal]
(&wtrans) a> sigRight=[new &signal]
(&wtrans) a> sound record sigLeft sigRight 1.5
Recording from: 'Mappeur de sons Microsoft', driver version 5.0 (manufacturer
id=1, product id=2)
Recorded 66150 Samples (264600 Bytes)
= '1'
```

Remark the name or driver version number of your input device may be different from what is indicated above

You can check the sampling frequency of the recorded signal by typing

```
(&wtrans) a> = 1/sigLeft.dx
= '44100'
```

Actually, the default behaviour of the `sound record` command is to record at CD quality (44100 Hertz, 16bits/sample). You can change this behaviour by specifying the recording quality. The various recording qualities are summarized in Table 2.1. With the 'custom' recording quality, you can (and must!) specify the sampling rate and number of bits/sample. For example,

'cd'	:	44100	Hertz, (at least)	16	bits;
'voice'	:	22050	Hertz, (at least)	8	bits;
'phone'	:	8000	Hertz, (at least)	8	bits;
'custom'	:	<sampleFreq>	Hertz, (at least)	<nBitsPerSample>	bits.

Table 2.1: The various recording qualities

to record 3 seconds of speech signal at 'phone' quality in signal '0', just type in

```
(&wtrans) a> sound record 0 3 phone
```

then speak in your microphone until three seconds have elapsed. You can display the recorded signal

```
(&wtrans) a> disp 0
```

or listen to it

```
(wtrans&) a> sound play 0
```

Remark If your sound input device does not support a given sound recording quality (for example if you ask 24 bit recording while it is not available), an error will be generated.

Remark It may happen that the sound recording quality you ask for is not *physically* available but that your sound input device driver can "emulate" it. For example, if you ask for a (custom!) sampling frequency of 92000 Hertz while your sound card only supports up to 48000 Hertz, the driver may provide LastWave with a recording which is obtained by resampling the digital data from your sound card.

Part II

Reference

Chapter 3

Package sound 2.0

Package allowing to deal with sounds. To learn about this package you should run the corresponding Demos. The read/write procedures have been taken from the 'sndfile' C-library made by Erik de Castro Lopo (erikd@zip.com.au). The sound package has a single command 'sound' that mainly allows : 1) to read/write sound files using different formats (mainly aiff, wav, au, raw and lw for LastWave file format obtained via the standard read/write commands), 2) to play a sound (on Unix computers you need to configure the system, please just type 'sound play' to learn about how to configure it and 3) to record a sound (this feature is available only on Macintosh and Windows computers.
*** Authors and Copyright : E.Bacry and R.Gribonval(for the package and the adaptation of the 'sndfile' library) and E. de Castro Lopo (for the original 'sndfile' library)*

3.1 Commands related to dealing with sounds.

- **sound** (in file `scripts/sound/sound.pkg`)
  - **sound** `format <matchExpr> [<format>]`

    If <format> is not specified, then it just lists all the available sound formats. If it is specified it defines a new format named <MatchExpr> corresponding to format <format>. Let us note that in the raw format names you can specify the sampling rate and the number of channels. For instance the format 'raw16_le_44100_1' refers to raw files using 16 bits integers, little endian, a sampling rate of 44000Hz and a single channel. In order to avoid writing very long suffices, you can rename them using this command. For instance, to rename that format

into a simple name such as 'mysnd', you should type 'format mysnd raw16_le_44100_1' (you should put it in your startup so that the format is defined at each runtime). Then you can use the '.mysnd' suffix.

- **sound** `info <filename>`

  Gets some information about the soundfile <filename>. It returns a listv made of <nbSamples> <samplingRate> <nChannels> <soundFormat>

- **sound** `play <sigL> [<sigR>] [<sampleFreq>] [-n]`

  WARNING : On Unix computers, there is no builtin command to play audio signals, hence you MUST perform some changes before using this command. In order to learn about the changes just type 'sound play'. On Macintosh and Windows (Cygwin) computers, it works just fine. This command plays a signal <sigL> in mono or a pair of (left/right) signals <sigL> <sigR> in stereo using the default sound output device. By default, a mono signal is normalized (between -1 and 1) before playing, and a stereo signal is jointly normalized (i.e. the relative level of the two channels is preserved). Option '-n' allows to avoid normalization (both <sigL> and <sigR> are then assumed to have values between -1 and 1, otherwise clicks will occur).If <sampleFreq> is not specified then the 'dx' field of <sigL> is used to compute the sample frequency, otherwise <sampleFreq> is used.

- **sound** `read <sigL> [<sigR>] <filename> [<rawFormat>] [-(s|S) <startingPoint>] [-(d|D) <duration>] [-n]`

  Reads the signals <sigL> and <sigR> from a sound file. You can specify a <sigR> only if the soundfile has more than 1 channel (such as a stereo file). You can specify the starting sample the signals should be read from and the duration that should be read using the options -d and -s (<startingPoint> and <duration> are expressed in seconds) or -D and -S (<startingPoint> and <duration> are expressed in seconds). If the option '-n' is not set then the maximum format range is renormalized to [-1,1] otherwise it is kept as it is. If the file is in raw format, and if you do not use the '-r' option it will try to use the extension of the <filename> (if there is one) in order to understand what the format is (i.e., number of bytes, endianness, sampling rate and the number of channels). If you want to specify the parameters of raw format files, you must specify the <rawFormat> to use (this is useless if the filename has an extension which specifies the format it is in).

- **sound** `record <sigL> [<sigR>] [<soundQuality>=cd [<sampleFreq> <nBitsPerSample>]]`

  WARNING :Works only on Macintosh and Windows (Cygwin) computers. Records a sound using default sound input device that supports the <soundQuality> and sets it in in <sigL> (mono recording) or <sigL> <sigR> (stereo recording). The recorded values are between -1 and 1 and the 'dx' field is set using the sample frequency. By default the 'quality' is 'cd'. The various possible qualities are
  'cd' : 44100 Hertz, (at least) 16 bits;
  'voice' : 22050 Hertz, (at least) 8 bits;
  'phone' : 8000 Hertz, (at least) 8 bits;
  'custom' : <sampleFreq> Hertz, (at least) <nBitsPerSample> bits.

- **sound** `stop`

  WARNING : Works only on Windows computers. Stop a currently playing sound.

- **sound** `write <sigL> [<sigR>] <filename> [<format>] [-n [<max>]]`

  Writes in file <filename> the sounds <sigL> (and <sigR> if specified). If <format> is not specified then the suffix of the <filename> is used to understand what the format is. In case of raw formats the suffix should also contain the sampling rate and the number of channels. This is done using the syntax <rawformat>_<samplingRate>_<nChannels>. Thus for instance a file named 'file.raw16_le_44100_1' is a raw file using 16 bits integers, little endian, a sampling rate of 44100Hz and a single channel. In order to avoid writing very long suffices, you can rename them using the 'sound format' command. Before being written in the file, the signals are normalized in order to use the whole dynamic range. If '-n' is set this is no longer true. The <max> parameter specifies what should be considered as the the maximum value. If <max> is not specified then it is taken to be the maximum available in the corresponding format (e.g., the range is [-32768,32767] if the format uses 16 bits integers).

3.2 Script Commands

3.3 Demos

Here is a list of all the Demo files and for each of them all the corresponding Demo commands. To try a Demo command, you should first source the corresponding Demo file then run the command. (When sourcing the Demo file, LastWave tells you about all the commands included in this file).
The Demo files corresponding to this package are :

Demo file **DemoSound**

• **DemoMPSound** (in file `scripts/sound/DemoSound`)

WARNING : YOU SHOULD FIRST RUN THE DEMO 'DemoSound' OF THE SOUND PACKAGE AND THE DEMOS 'DemoMPRegAlgo' and 'DemoMPFastAlgo' OF THE MP PACKAGE BEFORE RUNNING THIS DEMO. Demo command that performs the matching pursuit of a piano sound and that teaches you the sound capabilities of the 'mp' package. This demo involves computation that can involve 2-3 minute computations. This demo is also included in the 'mp' package. Warning : This demo uses a lot of memory, thus, if you are running on a Macintosh, you should allocate at least 20Mo to LastWave before running it.

• **DemoSound** (in file `scripts/sound/DemoSound`)

This is a demo that teaches you how to deal with sounds.
You should first run the 'DemoSignalDisp' demo.

Index