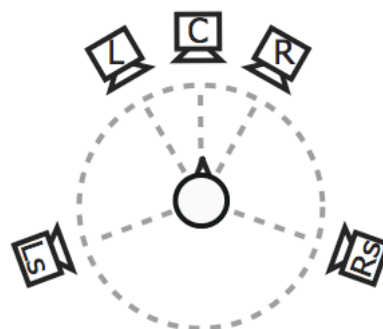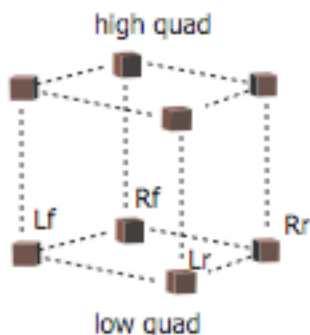# CDP Multi-channel Toolkit

abfdcode   abfpan   abfpan2   channelx   chorder   chxformat   copysfx   fmdcode
interlx   njoin   nmix   paplay   rmsinfo   sfprops

Introduction

The CDP Multi-Channel Toolkit comprises a suite of command-line programs (currently for Windows and OS X) to assemble and manipulate the channels of multi-channel soundfiles. It supports the standard PCM AIFF and WAVE file formats, with a special emphasis on Microsoft's WAVEFORMATEXTENSIBLE (WAVE_EX) file format[1]. This incorporates two features of special interest - the ability to associate one of eighteen named speaker positions with a channel (e.g. for 5.1 surround sound), and support for third-party custom file formats. In this respect the Toolkit supports and demonstrates the new AMB file format for Ambisonic B-Format surround audio[2]. An important use of the Toolkit is to assemble individual audio channels into a B-Format file for use with one of the many decoders now available as standalone applications or plugins[3].

This document does not comprise a tutorial on these basic aspects of multi-channel audio production; the reader is directed to the web links given in the footnotes. This is especially important with respect to B-Format audio. This is an immense and often complex subject (though simple enough in principle for users), which is not covered here in other than summary form.

Note that the focus of this Toolkit is on channel and format manipulation. With the exception of a demonstration program for B-Format panning, no audio processing (dsp) facilities are provided other than basic amplitude scaling. Readers interested in the general signal processing of multi-channel files (e.g. filtering, and a wide range of concrète-style processing) are encouraged to investigate the full CDP system[4]. A possible exception for a future release of the Toolkit is the addition of basic shelf filters and distance compensation for B-Format decoding.

Update, October 2009

A number of new programs are provided in this update, while others present since the original release have in many cases been revised with new features. A few of these programs are now deprecated in favour of new programs with more facilities. Deprecated programs may be omitted in future versions of the Toolkit.

With this release all programs support the maximum 4GB files sizes available in the WAVE and AIFF file formats.

OS X: the minimum OS version supported is 10.4 (Tiger). The programs are all Universal Binaries, tested under 10.5 (Intel) and 10.4 (G4).

New programs:

abfpan2   : pan sound around the listener using 2nd-order B-Format encoding.
chorder   : reorder channels of a file.
chxformat   : change the GUID or speaker position flags in a WAVE_EX file.
fmdcode   : decode 1st or 2nd order .AMB file to a number of standard speaker
                     layouts using the Furse-Malham and David Moore coefficients.
                  The coefficients for the 5.0 layout were computed by David Moore using
                  High-Performance Audio Computing (HiPAC) techniques.
rmsinfo   : Report RMS and average power of each channel in a file.

Deprecated programs:

abfdcode   : The original B-Format decoder, limited to quad output. Replaced by
                     fmdcode.
play24x   : The original native player program for Windows systems, using
                    DirectSound and WAVE_EX. Limited to quad B-Format decoding.
                    It is recommended to use the new expanded *paplay* program (based on
                    the *Portaudio* library) available for both platforms.

The Toolkit Programs

For the most convenient general use, the programs should be copied to a non system directory on the user's PATH. This documentation assumes the user is familiar with the general use of the command line (DOS console in Windows, Terminal in OS X) to run programs, navigate directories, employ shell commands, etc.   The programs are "destructive" in that repeating any command that creates an output file will replace the old file with the new one.
        When invoked with no arguments, all the programs display a detailed usage message.

- *abfdcode* : simple horizontal 1st-order B-Format decoder.
    Version: 1.1

Usage:
```
abfdcode [-x] infile outfile
         -x   :   write WAVE_EX (Quad) outfile format
               (requires .wav extension)
```

This program is now deprecated: use *fmdcode*.

- *abfpan* : apply fixed or orbiting 1st-order B-Format pan to mono infile.
  Version : 1.5.
  Usage:
  ```
  abfpan [-b][-x][-oN] infile outfile startpos endpos
       0.0 <= startpos <= 1.0 (0.0 and 1.0 = Centre Front)
     endpos: endpos < 0.0 gives anticlockwise rotation,
       endpos > 0.0 gives clockwise rotation.
       Units give number of revolutions, fraction gives final
               position.
       Set endpos = startpos for fixed pan.
    -b  = write output as horizontal Ambisonic B-format
           (std format)
    -x  = write B-Format (WAVE_EX) format file
       (requires .amb or .wav file extension)
       Default: write standard m/c soundfile.
    -oN = number of B-Format output channels: N = 3 or 4 only.
      Default: write 4-ch file.
  ```
  This program generates a simple periodic rotating pan around the listener, using 1st-order B-Format (horizontal) encoding. The start and end values determine the number of revolutions performed over the duration of the file. Identical values will result in a fixed radial position. Integer positions define centre front. A negative *endpos* value generates an anti-clockwise pan.
       Note that this program requires the −x flag to create a WAVE_EX file; this should always be used when writing a .wav format file. Use the −b flag to write the B-Format channels; both flags are needed to create an AMB format file. Such a file can then be decoded using *fmdcode* to a choice of speaker layouts.

- *abfpan2* : apply fixed or orbiting 2nd-order B-Format pan to infile
  Version: 1.0.
  Usage:
  ```
  abfpan2 [-gGAIN][-w] [-p[DEG]] infile outfile startpos endpos
      infile : mono source.
   outfile: 2nd order B-format output.
      0.0 <= startpos <= 1.0 (0.0 and 1.0 = Centre Front).
   endpos : endpos < 0.0 gives anticlockwise rotation,
          endpos > 0.0 gives clockwise rotation.
          Units give number of revolutions,
              fraction gives final position
          Set endpos = startpos for fixed pan
  -gGAIN : scale infile amplitude by GAIN (GAIN > 0).
  -w     : Write standard soundfile (wave, aiff)
          Default: WAVEX B-Format; use .amb extension.
  -p[DEG]: write full 9-channel (periphonic) B-Format file.
      Default: write 5-channel (2nd-order horizontal) file.
      DEG:  optional fixed height argument (degrees).
  ```

```
            Range = -180  to +180,
            where  -90 = nadir, +90 = zenith (directly above).
            Default: DEG=0;  height channels (Z,R,S,T)  will be empty.
```

This new extended version of *abfpan* employs 2nd-order B-Format encoding, with optional fixed elevation. Output is in AMB format; there is no decoded output option. A horizontal-only output comprises five channels, one with height ("periphonic") comprises 9 channels. Pan arguments are as for *abfpan*. Note that the default .wav output is in WAVE_EX format; use the -w flag to write a plain WAVE format file, which may be needed by some legacy applications.

- *channelx* : extract all channels from m/c file
  Version: 1.5
  Usage:
  ```
  channelx [-oBASENAME] infile chan_no [chan_no...]
    -oBASENAME = base name (with extension) of outfiles
          (appended *_cN for ch N)
  ```

This workhorse program extracts one or more channels (counted from 1) from the infile to mono output files. By default, outfiles are named as *infile_c1*, *infile_c2*, with the source file extension. Alternatively, use the −o flag to define a custom name.

- *chorder* : reorder soundfile channels
Version: 1.0
Usage:
```
chorder infile outfile orderstring
  orderstring = any combination of characters a-z inclusive.
  Infile channels are mapped in order as a=1,b=2...z=26
  (For example: channels in a 4-channel file are represented by the
    characters abcd; any other character is an error).
  Characters must be lower case, and may be used more than once.
  Duplicate characters duplicate the corresponding input channel.
  The zero character (0) may be used to set a silent channel.
  A maximum of 26 channels is supported for both input and output.
  NB: infile (WAVEX) speaker positions are discarded.
  The .amb extension is supported for the outfile.
```

New in this Update, this program expedites many of the tasks previously requiring a combination of *channelx* and *interlx*. Channels may be reordered while retaining the number of them. The program also supports both extracting a subset of input channels (e.g. 1st-order channels from a 2nd or 3rd order file), and augmenting the channel count by duplicating input channels and/or defining a number of empty (silent) channels. All .wav output files are in WAVE_EX format - the speaker positions (and/or the GUID) can be changed using *chxformat*.

Example 1 : convert anti-clockwise quad layout to standard surround quad:

```
chorder inregular.wav outquad.wav adbc
chxformat -s0x33  outquad.wav
```

Example 2: convert 5.0 surround file to 5.1 (with silent LFE channel):

```
chorder  in50.wav out51.wav abc0de
chxformat -s0x3f out51.wav
```

See also *fmdcode* below.

• *chxformat* : modify WAVE_EX header to change GUID and/or speaker positions.

WARNING: this program is *destructive* - it modifies the header of the infile. File system errors are always a danger. Do not use with files that cannot be replaced - use *copysfx* instead. It can also be used safely in test mode to report relevant header properties. In this release this program is marked as beta. While extensively tested (it employs full header parsing, it does not take any "short cuts"), as a destructive program it comes with no guarantees, nor warranty. Note that only the file header is modified, the audio data itself is untouched.

Version : 1.0 ()beta)
Usage:
```
chxformat [-m | [[-t] [-gGUID] [-sMASK]] filename
 -gGUID : change GUID type between PCM and AMB.
          Plain WAVEX: GUID = 1
          AMB:         GUID = 2
 -sMASK : change speaker position mask.
          MASK = 0:  unset channel mask
             else set to MASK
          MASK may be given in decimal or hex (prefix '0x').
 -m     : (not in combination with other options)
 NOTE: speaker positions are only supported for WAVEX PCM files.
      If GUID is set to 2, the -s option should not be used.
       Any existing speaker positions will be set to zero.
 Type chxformat -m to see list of WAVEX mask values.
 -t     : Test only: do not modify file.
         If only infile given, program prints current GUID type
          and speaker mask.
 In test mode, program checks file, prints current channel mask as
binary, and new mask in binary, if -s option set.
```

This program operates exclusively on files in the WAVE_EX format. Some familiarity with this format is advisable before using this program.

The -g option can be used to change a file from AMB format to .wav or *vice versa*, e.g. to be able to load a B-Format file into and editor or DAW that does not know about the AMB format. The program automatically assigns the appropriate GUID according to whether the samples are integer or floating-point.

The -s option is used to change the contents of the WAVE_EX speaker position flags, either to zero ("generic WAVE_EX) or to some other layout. The flag value can be entered in either hexadecimal form or decimal.

Use the -m flag option on its own to see the list of available positions, together with examples of the most common layouts.

Use the -t flag to see the current speaker layout of the file, and to verify the chosen speaker layout. The layout value is printed in binary, to facilitate checking against the position list.

Use the program with just a file name to see relevant properties of the file. For a comprehensive general format report use *sfprops*.

- *copysfx* : copy soundfiles; convert from one format to another.

Version: 1.9

Usage:

```
copysfx [-d][-hN][-sN][-tN] infile outfile
-d   : apply TPDF dither to 16bit outfile.
-s   : force output sample type to type N.
      Available sample types:
      1  : 16bit integers (shorts)
      2  : 32bit integer  (longs)
      3  : 32bit floating-point
      4  : 24bit integer 'packed'
          (WAV only: AIFF/AIFC written as type 4)
      Default: format of infile.
-h   : write mimimum header (no extra data).
        Default: include PEAK data.
-tN  : write outfile format as type N
Possible formats:
0   : standard soundfile (.wav, .aif, .afc, .aifc)
1   : generic WAVE_EX (no speaker assignments)
2   : WAVE_EX mono/stereo/quad(LF,RF,LR,RR) - infile nchans must match
3   : WAVE_EX quad surround (L,C,R,S)       - infile must be quad
4   : WAVE_EX 5.1 format surround           - infile must be 6-channel
5   : WAVE_EX Ambisonic B-format (W,X,Y,Z...)- extension .amb supported
6   : WAVE_EX 5.0 Surround                  - infile must be 5-channel
7   : WAVE_EX 7.1 Surround                  - infile must be 8-channel
8   : WAVE_EX Cube Surround                 - infile must be 8-channel
default in all cases: outfile has format of infile
NB: types 1 to 8 are for WAV format only
```

This is the primary workhorse program in the Toolkit for copying and converting multi-channel files.

NEW in this version:
- -d flag to apply 2-LSB triangular dither to 16bit files.
- Support for 7.1 and cube speaker layouts.

REMOVED in this version:
- The sample container options 20bits in 24, and 24bits in 32. I have no evidence of anyone using or needing these (they were provided mainly to demonstrate aspects of the WAVE_EX format). Should I be informed to the contrary, these options can be

restored (or just use an older version of *copysfx*).
- -i flag (interpret 32bits as floats). Supported the original Cool Edit.
  Should now be obsolete.

Notes on usage:

By default the program writes a PEAK chunk to the output file. This stores the position and value of the first absolute maximum sample in each channel, together with a timestamp. Almost all the Toolkit programs print the PEAK data to the screen on completion; it can also be shown via *sfprops*. For integer formats, the value will of necessity be clipped to 1.0; for floating-point formats the value reflects the true peak values in the file.

However, it is still common for many applications to assume a fixed length header, and be broken by any file with such additional chunks. To pander to such feeble applications, use the −h flag to write a "minimal" header with no extra chunks.

Note that while the new program *chxformat* can save considerable disk space and time by changing the header of a file directly, it is inherently risky. *Copysfx* is non-destructive, and if space and time permits is still the recommended tool for converting files from one format to another. *Chxformat* may however be required where it is desired to set an unusual or experimental speaker position mask not supported by *copysfx*.

- *fmdcode* : decode 1st or 2nd-order B-Format file to a choice of speaker layouts.
Version: 1.0 beta.
Usage:

```
fmdcode [-x][-w] infile outfile layout
     -w    :   write plain WAVE outfile format
                (.wav default - use generic wavex format).
     -x    : write std WAVEX speaker positions to header
              (applies to compatible layouts only;
                requires .wav extension).
  layout    : one of the choices below.
     Output channel order is anticlockwise from centre front
          except where indicated.
  Layouts indicated with * are compatible with WAVEX
               speaker position order.
  Available speaker layouts:
  1    : *   mono (= W signal only)
  2    : *   stereo (quasi mid/side, = W +- Y)
  3    :     square
  4    : *   quad FL,FR,RL,RR order
  5    :     pentagon
  6    : *   5.0 surround (WAVEX order)
  7    : *   5.1 surround (WAVEX order, silent LFE)
  8    :     hexagon
  9    :     octagon 1 (front pair, 45deg)
  10   :     octagon 2 (front centre speaker)
  11   :     cube (as 3, low-high interleaved. Csound-compatible.)
  12   : *   cube (as 4, low quad followed by high quad).
```

With the exception of layouts 6 and 7, this program implements the widely used Furse-

Malham decoding coefficients for 1st and 2nd order B-Format[5]. These coefficients are also employed in the *bformenc1* and *bformdec1* opcodes for Csound. Layout option 11 (cube) corresponds to the one employed by *bformdec1*, while layout option 12 is (approximately) WAVE_EX-compatible.

In all cases (and as in Csound) the "controlled opposites" (also known as "in-phase") decoding solutions are employed, without either shelf filters or distance compensation. These features may appear in a future release. These solutions apply to *regular* geometric layouts - that is, where speakers are evenly distributed around the circle:

Type 3 : square   Type 5: pentagon   Type 6: hexagon
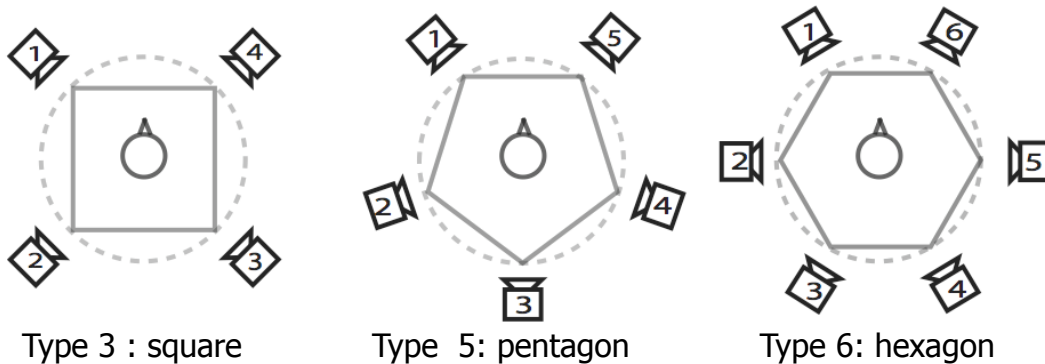
In options 9 and 10, two alternative octagonal layouts are defined, depending on the use of either one speaker at centre front, or two left and right of centre front:

Type 9 : octagon 1   Type 10: octagon 2
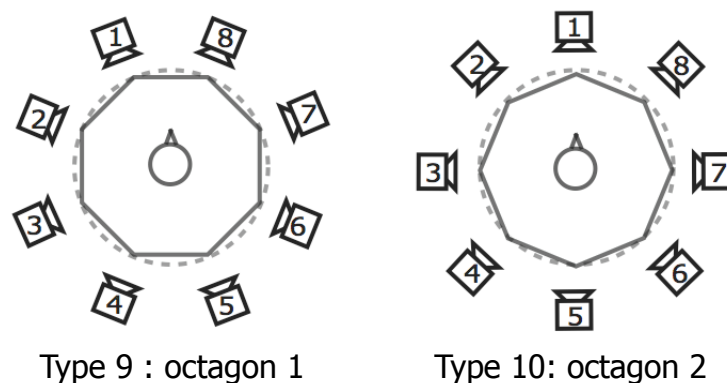
As the figures show, in all these cases the outfile channels (speaker feeds) are ordered anti-clockwise relative to centre front. The opposite order is also widely demonstrated in the literature; should this order be required, the program *chorder* can be used to reverse the sequence.

The principle of B-Format encoding and reproduction is predicated on such regular layouts, which extend to the periphonic case ("with-height" reproduction) using shapes such as the simple cube and the dodecahedron. The Toolkit does not currently address layouts involving more than eight speakers.

WAVE_EX-compatible speaker layouts.

As the usage message indicates, the majority of these regular layouts are not compatible with the WAVE_EX speaker positions. Despite the fact that they are described by name rather than by precise co-ordinates, their positions are in general *assumed*, e.g. by reference to standard irregular layouts such as the ITU 5.1 surround layout (see right-hand figure at the top of this document). The order of the speaker position flags in WAVE_EX is based on alternating left/right pairs (e.g. layout 4 - rectangular quad). We can use the applicable WAVE_EX positions to represent this layout only with the understanding that the front pair are spaced to subtend not the conventional stereo 30-degree angles left and right of centre front, but 45-degree angles, such that the four speakers form an exact square. With the same understanding, the cube layout type 12 can also be represented in WAVE_EX, as that includes support for "TOP" positions matching the usual "LOW" positions. However even here we must take care - the purely horizontal layouts naturally address speakers arranged at head height. In contrast (and as with all such layouts), the B-Format cube layout places the listener in the *centre* of the cube, so that speakers are equidistantly *below* and above the listener. To achieve such a layout in a practical context, these low speakers are typically placed at floor level.

As WAVE_EX does not define much in the way of regular layouts, by default the program sets the speaker position flags to zero. This signifies a "generic" layout, in which the channels are delivered to the soundcard in simple ascending order; it is then up to the user to route those outputs appropriately. One advantage of this approach is that it bypasses attempts by the operating system to re-map the signals to whatever layout it thinks you have! To write such layouts with the appropriate WAVE_EX speaker mask, use the -x flag. Advanced users can use *chxformat* to experiment with speaker masks for the other regular layouts. Attempting to play such files directly using standard Windows players such as *Media Player* is very likely to give unintended results. See the description of *paplay* for more on this issue.

5.1 layouts - types 6 and 7.

As noted above, these layouts are not derived from the Furse-Malham specifications. The coefficients for the 5.0 and 5.1 layouts (using the so-called "velocity" decode option) have been kindly supplied by David Moore[6] as a result of recent research employing advanced High-Performance Audio Computing (HiPAC) techniques[7]. They are a result of a research project employing advanced high-performance computing techniques (HPC) to find optimal coefficients through the use of exhaustive search algorithms.

For B-Format decoding, the LFE (low-frequency effects) channel (the '1' of '5.1') is not used. Accordingly, two decode options are provided, supporting five and six-channel files. The latter thus includes an empty LFE channel. When using the WAVE_EX speaker option (−x flag) you may need to use the six-channel option to ensure your OS (and hardware) can render the file correctly.

Where possible, a second-order B-Format source should be employed; this is generally held to decode better to the ITU layout than first-order. For casual exploration of B-Format surround, this layout appeals as it corresponds to the expected speaker layouts for home cinema. Note however that for best results the speakers should be identical (or at least very closely matched) and full-range. For a simple test, *abfpan2* can

be used to pan a sound around the circle, using 2nd-order encoding.

- *interlx* : interleave two or more files to make a multi-channel file.
Version: 2.0
Usage:
```
interlx [-tN] outfile infile1 infile2 [infile3...]
    Up to 16 files may be interleaved.
    Output format is taken from infile1.
    Files must match sample rate and number of channels,
        but can have different sample types.
    To create a silent channel, for infile2 onwards,
       use 0 (zero) as filename. Infile1 must be a soundfile.
    NB: Speaker-positions in WAVE_EX infiles are ignored
    Note that the same infile can be listed multiple times,
        for example, to write a mono file as stereo, quad, etc.
    The .amb B-Format extension is supported:
        the program warns if channel count is anomalous.
    Known Bformat channel counts: 3,4,5,6,7,8,9,11,16.
-tN  : write outfile format as type N
    Available formats:
    0 : (default) standard soundfile (.wav, .aif, .afc, .aifc)
    1 : generic WAVE_EX (no speaker assignments)
    2 : WAVE_EX mono/stereo/quad(LF,RF,LR,RR)- infile nchans must match
    3 : WAVE_EX quad surround (L,C,R,S)      - infile must be quad
    4 : WAVE_EX 5.1 format surround          - infile must be 6-channel
    5 : WAVE_EX Ambisonic B-format (W,X,Y,Z...) - .amb supported
    6 : WAVE_EX 5.0 surround                 - infile must be 5-channel
    7 : WAVE_EX 7.1 Surround                 - infile must be 8-channel
    8 : WAVE_EX Cube Surround                - infile must be 8-channel
    In all cases: outfile has sample format of infile1
    NB: types 1 to 8 are for WAV format only
```

NEW in this version:
    format options 7 and 8 (7.1, cube).
REMOVED in this version:
    -x flag -: redundant.

This program complements *channelx*, and is the primary workhorse tool for assembling multi-channel files from multiple input files. As the usage message shows, it serves two key tasks - to create WAVE_X files in particular formats, and to create AMB B-Format files from mono input files comprising individual B-Format signals. For the latter task, use -t1 to make a generic WAVE_EX file - no speaker positions are defined for the AMB format.

- *njoin* : concatenate multiple soundfiles, with optional CUE list output for CD burning.
Version: 1.0
Usage:
```
njoin [-sSECS | -SSECS][-cCUEFILE][-x] filelist.txt [outfile]
      filelist.txt: text file containing list of sfiles
                     in order. One file per line.
                     Channel spec (if present) must be the same,
                     but files with no spec assumed compatible.
```

```
     -cCUEFILE   : if outfile used, generate cue textfile as CUEFILE.
     -sSECS      : separate files with silence of SECS seconds
     -SSECS      : as above, but no silence before first file.
                   Default: files are joined with no gap.
    -x           : strict: allow only CD-compatible files:
                   Must use sr=44100, minimum duration 4 secs.
     NB: Files must match sample rate and number of channels,
      but can have different sample types.
     Output sample format taken from file with highest precision.
     If no outfile given: program scans files and prints report.
```

NEW in this version:
-x flag to enforce strict CD-compatible CUE file
OS X: use of the ~/ notation for the user's home directory is supported in *filelist*.

This utility program stands somewhat apart from the others in that it is only indirectly concerned with multi-channel files. Its primary purpose is simply to concatenate multiple files into one output file. Channel counts and sample rates must match. Secondarily the program supports preparation for CD-R burning by means of basic support for the standard CUE list file, used by a number of CD burning applications. Files can be separated (prefixed) by a defined amount of silence.

      The -x flag can be used to enforce strict CD compatibility in the filelist: the sample rate must be 44100, files must be stereo, and individual soundfiles must be at least four seconds long. The filelist is a text file with one complete file path per line.

      The program exits with an error message if the total duration exceeds the 4GB capacity of the file format. When used without an outfile argument, the program computes the total duration and reports to the console.

- *nmix* : simple mix of two multi-channel files, with optional offset.

Version: 2.0

Usage:
```
nmix [-d][-f][-oOFFSET] infile1 infile2 outfile
     -d       : apply TPDF dither (16bit format output only).
     -f       : set output sample type to floats.
                 Default: outfile type is that of infile 1
     -oOFFSET:  start infile2 at OFFSET seconds.
Files must have the same channel count.
WAVE-EX files must have the same speaker layout
```

NEW in this version:
  -d : apply TPDF dither to 16bit outputs
  -f : make floats outfile.

Nothing much to say about this program - it mixes two soundfiles together. The channel counts (and speaker layouts if any) must match.

- *rmsinfo* : scan file and report rms and average power level statistics.

Version: 1.0

Usage:

```
rmsinfo [-n] infile [startpos [endpos]]
Standard output shows:
    RMS level
    Average level
    DC level (bipolar average)
-n : Include equivalent 0dBFS-normalised RMS and AVG levels.
Optional arguments:
startpos  :  start file scan from <startpos> seconds.
endpos    :  finish file scan at <endpos> seconds.
To stop a scan early, use CTRL-C.
Program will report levels up to that point.
```

This new program complements *sfprops* by determining the overall power level (loudness) of each channel of the infile. It uses both RMS (Root-Mean-Square) and raw average (normalised sum of all samples) computations. In addition, it displays the average level of any DC present. All levels are calculated relative to digital peak (0dBFS). Thus a full-range sinusoid will have a reported RMS level of -3dB.

Use the -n flag to see the equivalent levels if the file is normalised to 0dBFS.

For sustained sounds, the RMS and average levels will be very similar. For percussive, speech-based, and other sounds with widely differing amplitudes, they may be markedly different (the average level will be lower than the RMS level). In such cases, the information reported gives an overall indication of the "peakiness" of a file, and may be a guide to the choice of compressor thresholds.
        Note that for *static* B-Format soundfields (i.e. no motion of sources) the information will demonstrate the standard -3dB reduction of the W signal (as employed in all the Toolkit programs, and in keeping with the Furse-Malham conventions). For mobile sources (e.g. from using *abfpan2*) the levels may well be reported as equal, depending on the overall bias (if any) of the panning (the final reported value represents the average over the whole file). Other conventions are under consideration within the surround sound community for B-Format streams - in particular, normalisation schemes that avoid the traditional 3dB reduction of W. This will in principle lead to RMS values for W that are closely similar to those of other channels.
        The program recognises that it can take a long time to scan a large file. The optional arguments *startpos* and *endpos* can be used to select a section of a file to be scanned. Also, CTRL-C can be used to terminate the program, in which event the levels up to that point will be displayed.

* *sfprops* : report properties of a soundfile
Version: 2.0
Usage:
```
    sfprops infile
```

NEW in this version:
* identifies 7.1 and cube speaker layouts
* PEAK chunk report includes level report in dB.

This is a purely informative program: in addition to reporting the standard properties of a

soundfile it gives details of any WAVE_EX speaker positions and identifies the layout where one of the standard ones (as supported by other Toolkit programs). It prints all PEAK information if present. Use *chxformat* to find low-level information on the WAVE_EX speaker mask.

Playback of multi-channel files.

- *paplay*.

Version: 2.0
Usage:
```
paplay [-dN][-gN][-i][-l][-b[N]] soundfile [from]
        -dN  : use output Device N
        -gN  : apply gain factor N to input.
        -i      : play immediately (do not wait for keypress)
        -l      : loop file (continuous repeats from start-time to end)
        from   : set start time (secs)
        -b[N]   : apply 1st-order B-Format decoding to std soundfile
                : (file must have at least 3 channels)
                  B-Format files (.amb) will be decoded automatically.
                N sets speaker layout to one of the following:
                1   : *  mono (= W signal only)
                2   : *  stereo (quasi mid/side, = W +- Y)
                3   :    square
                4   : *  quad FL,FR,RL,RR order   (default)
                5   :    pentagon
                6   : *  5.0 surround (WAVEX order)
                7   : *  5.1 surround (WAVEX order, silent LFE)
                8   :    hexagon
                9   :    octagon 1 (front pair, 45deg)
               10   :    octagon 2 (front centre speaker)
               11   :    cube (as 3, low-high interleaved.(as Csound)
               12   : *  cube (as 4, low quad followed by high quad).
```

NEW in this version:
- Extended set of B-Format decode layouts (as in *fmdcode* q.v.).
- −g flag to scale level of infile
- Windows: supports ASIO as well as DirectSound.
- Various internal fixes and enhancements
Windows:
- -x flag available to send the file's speaker mask to the hardware, The default is to send a zero mask, so that channels map directly to successive hardware outputs.

The same B-Format decoding is employed here as in *fmdcode*. As in that program, no shelf filters or nearfield compensation are employed.

When *paplay* is invoked without arguments, it displays the usage message followed by a list of the available audio devices. Note that input and output devices are listed separately, so that the list can become quite long! The current default output is indicated by a leading

asterisk before the device number.

The -i flag is provided primarily to enable the program to be invoked from a scripting languge (e.g. tcl/tk, Matlab/Octave, etc) where it is inconvenient to interact with the program via the keyboard. Otherwise, note that when launched, the program waits for a keypress to commence playback. As the first block of data is pre-loaded, playback will effectively start immediately.

Testing audio routing

It is strongly recommended that users test their routing before trying to play a multi-channel file. The Toolkit can itself be used to prepare suitable test materials. One approach is to create a special file (e.g. of spoken idents, or clearly distinguishable tones), with each channel clearly identified in sequence. The -l flag can be used to loop this or any file until stopped via CTRL-C, while any necessary adjustments to routing or levels are made. For B-Format playback, it is especially important that all speakers are as closely matched in level and distance as possible, and that any mixer or other level controls are ganged together.

Such a file is especially easy to create using Audacity : record each ident in sequence as a new track; then use "Export" with the "Advanced Mixing Options" enabled, to save each track as a channel of a multi-channel file. You can use *copysfx* to convert this file into the required WAVE_EX format. Alternatively, use the "Export Multiple" option to save each track as a separate numbered file, and create the final multi-channel file using *interlx*. This approach is indicated, for example, when the same idents can be used, in different combinations, for a variety of multi-channel layouts.

To check that speaker levels are exactly matched, a file with test signals is required, with (ideally) a loudness meter at the listening position. This is especially important if unmatched speakers (and amplifiers) are employed!

OS X.

It goes without saying that *paplay* uses CoreAudio. OS X itself recognises both WAVE_EX files and the most standard speaker layouts and renders them (e.g. via Quicktime) appropriately given multi-channel hardware. *paplay* simply sends the channels in order to the chosen device.

The historical limitations (and bugs) with respect to the WAVE formats associated with earlier Macintosh systems are long gone, and with the move to Intel processors the case for using WAVE (in general) over the AIFF formats is even stronger. The AIFF format is now arguably all but obsolete - such definitions as exist in the AIFF and AIFF-C specifications for multi-channel files are variously obsolete and ambiguous. Apple' s replacement for the AIFF family is the new CAF file format; support for this may appear in later updates to the Toolkit.

Windows

This version of *paplay* supports both the DirectSound and ASIO APIs. Accordingly, the displayed list of devices will longer, with the same hardware identified for each API as

appropriate.

Note the -x flag exclusive to the Windows platform. Windows recognises a very limited set of standard speaker layouts ; these can be selected (in this example, in XP) via the "Advanced" button on the Audio page of "Sound and Audio device Properties". To play a multi-channel WAVE_EX file using Media Player or other Windows-standard player, users will need to check which layout is selected - if there is any mismatch (such as trying to play a 5.1 file to stereo speakers), the Windows kernel mixer will attempt to map the audio channels to that layout in some representative way. This has some virtues; e.g. in being able to audition an arbitrary multi-channel file over stereo speakers. Many commercial editors and DAW perform in much the same way, sending the WAVE_EX channel mask directly to the subsystem. Should the user want to replicate this behavior using *paplay*, the -x flag can be used. This flag is not available in the OS X version of *paplay*.

Note however that when Windows plays, for example, a quad file to a stereo device, the rear channels are typically rendered at a lower level.

For such tasks as B-Format decoding to relatively unorthodox layouts (such as hexagon, octagon), this automatic remapping is not useful beyond a simple integrity check. The default behaviour of *paplay* is to send a "generic" mask value of zero to the audio subsystem, so that channels are simply mapped to successive device outputs.

[1]**Notes**

See for example, http://dream.cs.bath.ac.uk/researchdev/wave-ex/wave_ex.html. Microsoft's document describing the format is at: http://www.microsoft.com/whdc/device/audio/multichaud.mspx.

[2] http://dream.cs.bath.ac.uk/researchdev/wave-ex/bformat.html and http://www.ambisonic.net/(see the link " Getting Ambisonics Around").

[3] Many listed at the above references, via http://www.ambisonic.net/

[4] http://www.composersdesktop.com/

[5] http://www.muse.demon.co.uk/ref/speakers.html

[6] See http://compeng.hud.ac.uk/sengdjm. His approach is described in his paper for the 126th AES Convention (Munich, 2009), *The Potential of High Performance Computing in Audio Engineering*. See: http://www.aes.org/events/126/libraries/session.cfm?code=P14

[7] See http://dream.cs.bath.ac.uk/HiPAC/index.html. The Toolkit web page will give (or point to) more details as they become available.