CDP System Utilities

(with Command Line Usage)

All functions in this group are standalone programs.

Utility Functions

ALIAS
Create a shortcut to a soundfile (PC only)
CDPCONV
Utility to convert analysis files from PPC to Intel format (MAC only)
COLUMNS
Manipulate or generate columns of numbers
COPYSFX
Copy/convert a soundfile
DIRSF
Soundfile directory listing
GETCOL
Extract a column of numbers from a textfile
LISTAUDEVS
List audio devices
PAPLAY
Playback of multi-channel soundfiles
PUTCOL
Place a column of numbers into a textfile
PVPLAY
Play back (audition) an analysis or soundfile
RECSF
Record, creating a soundfile
VECTORS
Numerical operations between two columns of figures

ALSO SEE:
HOUSEKEEP Group
Housekeeping utilities for soundfiles
MULTI-CHANNEL TOOLKIT, from which COPYSFX and PAPLAY are taken.

ALIAS – Create a shortcut to a soundfile (PC only)

Usage

alias *full_path name*

Parameters

full_path – the location of the soundfile, specifying the drive, the directory (with subdirectories) in which it resides, and its name. E.g.,

```
d:\compositions\project6\whirr.wav
```

name – the name of the shortcut soundfile; it may be the same as or different from that of the original name.

Example command line to create an alias:

```
alias d:\compositions\project6\whirr.wav    whirr.wav
```

Understanding the ALIAS Function

A 'shortcut' on a PC system (Windows) is a path to a file. It does not duplicate that file. This can be particularly useful for soundfiles, which can be very large. If you create a shortcut to a soundfile, you can then access that file from your current directory simply by entering the alias name.

ALIAS creates a shortcut in the current directory. It can be treated as if it were an actual soundfile in the usual way.
To be recognised by CDP programs the alias *name* must include the extension (e.g. .wav), as shown in the above example.

In **DIRSF**, a soundfile for which a shortcut has been created is shown with its additional **.lnk** extension: e.g., **whirr.wav.lnk**.

NOTE: At the present time, ALIAS has NOT been included in the Release 7 distribution. (R.F.)

End of ALIAS

CDPCONV – Utility to convert analysis files from PPC to Intel format (MAC only)

Usage

cdpconv *infile* [*outfile*]

Provide an *outfile* name to perform a non-destructive copy. Otherwise the 'destructive' mode is used.

Example command line to carry out the conversion :

`cdpconv splash.ana splashcnv.ana`

Parameters

infile – input CDP file created on a PPC Mac, which can be any of **.ana**, **.frq**, **.trn**, **.evl**, **.fmt**, **.for**
outfile – optional new file, omitted if the new file is to replace the old one ('destructive')

Understanding the CDPCONV Function

Computer memory architectures fall into two types, according to the "endianness" order in which they write multi-byte numbers into memory. A Power-PC (PPC) based Macintosh uses a "big-endian" byte order, while a machine based on an Intel CPU (i.e. including an Intel-based Mac) uses the opposite "little-endian" order.

CDP analysis files employ a number of "custom" property fields written into the header. For historical reasons these do not maintain a specific endianness, but are simply written according to the "native" endianness of the host machine. **To transfer such files from one architecture to another, such as when moving from an older PPC Mac to an Intel Mac or PC, these property fields need to be 'byte-swapped'.**

You can determine if a file needs byte-swapping by running one of the CDP programs that reports file properties:

`sndinfo props` (may report failures to read the file)
`sfprops` (will report all properties, with bad values)

Typically, integer fields will have huge or negative values, and floating-point fields will likely appear as zero.

CDP provides this utility program, CDPCONV, to accomplish this task. It was written by Richard Dobson. It can be used *either* **destructively** (overwrites the file header directly) or **non-destructively** (creates a new file). Clearly the 'destructive' mode will be very fast, saves disk space and avoids the inconvenience of renaming files, but in theory always carries the risk, however unlikely, of some hardware error causing the file header to become damaged. The audio data itself is not touched at all.

Musical Applications

While one may often need to convert a single file at a time, it may be more convenient to carry out the conversion on those in a whole directory at once. The following information about a shell script describes how to do this.

Batch processing via a Shell script
OS X supports simple shell-based scripting via Terminal. This can be used to batch-process all files in the current directory. The example below (copy to a file called e.g., *anaconv.sh*) runs CDPCONV on all soundfiles with the **.ana** extension:

```
#!/bin/bash

read -n1 -sp "ANACONV: press any key to continue..."

for filename in *.ana
do
   echo
   if(test -a "$filename")
    then
      echo converting "$filename"
      cdpconv "$filename"
   fi
done
```

Note that it asks for a keypress before continuing – this gives you a chance to bail out if the script is run by accident!

To convert all CDP analysis files replace the line starting `for filename` with

`for filename in *.ana, *.frq, *.trn, *.evl, *.fmt, *.for` You can modify the search list in this way to select any required set of files.

Note that CDPCONV does not test the properties in any way – it simply performs the byteswap. Applying CDPCONV again to the same file will restore the original 'endianness'.

End of CDPCONV

COLUMNS – Manipulate or generate columns of numbers

Usage

columns *infile* [*outfile* [**-flag@**] [{*threshold*|}*threshold*]
ALSO:
columns *inoutfile1 infile2* [*infile3....*] **-J** to juxtapose columns in one file
columns *infile1 outfiles* **-S***X* to separate columns to *X* distinct files
columns *infile1 outfiles* **-N***X* to separate values to *X* distinct files, in blocks
columns *infile1 outfiles* **-Nr***X* to separate values to *X* distinct files, in rotation

For more information:

columns -f to see all available flags
columns -g for generative operations
columns -l for list manipulation and editing
columns -m for mathematical operations
columns -M for musical operations
columns -R for randome operations

Parameters

infile – input text file containing a column of numbers
outfile – optional output text file
-flag@ – flag to specify the operation to be performed on the input column of numbers. The '@' symbol indicates a numeric value to be supplied by the user, and may be optional (in square [] brackets).
'{' = less than: '}' = greater than
threshold is a numeric value and can only be used with **R**, **a**, **m**, **d**, **P**, **c** – the specified operation then applies only to values > or < the *threshold* value.

Understanding the COLUMNS Function

Although there are many operations available in COLUMNS, using the program is relatively straightforward. For example, the command

columns *infile outfile* **-flag@**

```
columns  acol.txt acoladd.txt -a20
```

will add 20 to every value in *acol.txt* and write the resultant column to *acoladd.txt*.

This is what the two files would look like:

```
acol.txt acoladd.txt
1        21
2        22
3        23
4        24
5        25
```

What can sometimes be confusing is how COLUMNS handles the first value in the list and whether or not the sums accumulate. You are recommended to perform as simple a test as possible on an operation you wish to use and study the result to see exactly what calculation(s) COLUMNS has made.

COLUMNS is fully implemented in *Sound Loom* as the **TABLE EDITOR** and in *Soundshaper* as the **DATA EDITOR**. There is excellent documentation on the **Data Editor** in 'Help > Soundshaper Help > DATA EDITOR'.

More information and user feedback on this utility would be welcome.

ALSO SEE: PUTCOL, **GETCOL**, **VECTORS** and **SNDINFO UNITS**, which is designed as a 'music calculator'.

End of COLUMNS

COPYSFX – Copy/convert soundfiles

Usage

copysfx [**-d**] [**-h***N*] [**-s***N*] [**-t***N*] *infile outfile*

Parameters

infile – a **wav** or **aif** soundfile
outfile – a **wav** or **aif** soundfile.
Default in all cases: *outfile* has the format of the *infile*.

-d – apply TPDF (2-LSB triangular) dither to a 16-bit *outfile*
-h*N* – write minimum header:

- **0** = minimum (no extra data: this avoids incompatibility with software which cannot read the extra data)
- **1** = PEAK data only
- Default (flag not used): include PEAK data.

-s*N* – force output sample type to type *N*

Available sample types:

1. 16-bit integers (shorts)
2. 32-bit integer (longs)
3. 32-bit floating-point
4. 24-bit integer 'packed'

Default (flag not used): format of *infile*

-t*N* – write *outfile* format as type *N*

Possible 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* number of channels 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, A ...) – the **.amb** extension is 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
9. – WAVE_EX 6.1 Surround – *infile* must be 7-channel

NB: Types 1 to 8 are for WAV format only.

About COPYSFX

COPYSFX is the primary workhorse program in the **MULTI-CHANNEL TOOLKIT** for copying and converting files from one format to another.

By default COPYSFX writes a PEAK chunk to the output soundfile. 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.

It is still common for many applications to assume a fixed length header, and be broken by any file with additional chunks such as the PEAK data. For such applications, use the **-h** flag to write a "minimal" header with no extra chunks.

Note that while **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 permit, is still the recommended tool for format conversion. CHXFORMAT may however be required where it is desired to set an unusual or experimental speaker mask not supported by **COPYSFX**.

This function is also documented in the **Multi-Channel Toolkit** manual.

End of COPYSFX

DIRSF – Soundfile directory listing

Usage

(dirsf -h or **dirsf -?** displays the Usage)

'**dirsf**' on its own provides a full listing of sound data in the current directory with any of the following extensions**:**

- soundfiles (**.wav** or these varieties of **.aif: .aif, .aiff, .aifc, .afc, .aic**)
- analysis files (any extension is acceptable, **.ana** is the convention)
- envelope files (**.evl**) for binary files containing amplitude envelope data (**.env** is no longer used)
- formant files (binary files for which the system-provided extension is **.for**) (**.fmt** is no longer used)
- binary pitch data files (**.frq**), i.e., as produced by REPITCH GETPITCH
- binary data transposition files (**.trn**), i.e., as produced by REPITCH COMBINE
- soundfiles with a shortcut (**.lnk** is appended)
- breakpoint files (**.brk** by convention – or **.txt**)

Other text files such as the note data file used in TEXTURE (i.e., ASCII text files) can be displayed with the computer's normal **dir** (PC) or **ls** (MAC OS X) command.

dirsf *filename.extension* provides information for the named soundfile.

Note that the extension must be specified. This is because the same filename may have different extensions: e.g., *owl.ana* (analysis file) and *owl.frq* (binary pitch data file – the pitch trace – derived from that analysis file).

Parameters

-l – list names only
-t – list modification dates and times
-d*path* – list only files in directory *path*
path – location of the soundfile in the directory system
filespec – filenames specified with wildcards, e.g., s*.ana

Understanding the DIRSF Function

The full soundfile display page shows, from left to right:

Bytes
the size of the soundfile in bytes
Channel
the number of channels. If the file is an analysis file, the number of channels shown is actually the number of filter channels in the analysis + 2 for header information.
Sample
the numerical format of the data (shorts or floats). CDP now supports the floating point soundfile standard, for greater fidelity, so a 'float' file can be either a soundfile or an analysis file.

Rate
: the sample rate of the soundfile, i.e., the number of times per second the source sound has been 'sampled'. CDP supports the following sample rates: 96000 , 48000, 44100, 22050, 32000, 16000. If the file is an analysis file, the number of channels shown is actually the frame rate, i.e., the number of analysis frames in the file. The PVOC manual explains how this works.

Seconds
: the duration of the soundfile in seconds

fmt
: the format of the file: **.wav** or **.aif** etc. (whether a soundfile or an analysis file with .ana extension), or **?** if not recognised by the system

Name
: the name given to the soundfile

Example DIRSF display

Bytes	Channel	Sample	Rate	Seconds	fmt	Name
785408	1	short	44100	8.905	W	softspoken.wav
6357096	1026	float	172	9.006	W	rmstr.ana

DIRSF also handles and displays information about multi-channel file types. Also see CDP MCTOOLS.

End of DIRSF

GETCOL – extract a column of numbers from a textfile

Usage

getcol *in_textfile out_textfile colno* [*skiplines*]

Parameters

in_textfile – input textfile from which to extract the column
out_textfile – output textfile to which to write this column
colno – integer to specify which column to extract (count starts at 1)
skiplines – number fo lines to skip from head of file (each blank line counts as one line)

Understanding the GETCOL Function

GETCOL provides a mechanism with which one can extract a column of numbers from a text file containing two or more columns and save it to a separate file, for processing (or printing, etc.). Using PUTCOL, it can later be placed in another file or returned to the source file.

Musical Applications

The COLUMNS program provides many ways of manipulating data, usually as single columns. For example, you might use it to quantise MIDI pitches. GETCOL can extract a column from a breakpoint file to provide an input file for COLUMNS.

ALSO SEE: PUTCOL, COLUMNS, VECTORS

End of GETCOL

LISTAUDEVS – list available audio devices

Usage

listaudevs

Parameters: none

About LISTAUDEVS

Listaudevs provides a list of the audio devices (input and output) on your computer. It reports to the console, but you can redirect the output to a text file by typing **listaudevs >*filename*.txt** (PC) or ....(OS X).

The format of the output list is amost identical to that produced by PAPLAY or PVPLAY (when invoked without parameters). Input and output devices are listed separately and the current default output is indicated by a leading asterisk before the device number. See examples in **PAPLAY** and **PVPLAY** below.

End of LISTAUDEVS

PAPLAY – Playback of multi-channel soundfiles

Usage

paplay [**-B***N*] [**-d***N*] [**-g***N*] [**-h***N*] [**-i**] [**-l**] [**-b**[*N*] | **-m**[*S*]] [**-u**] [**-x**] *soundfile* [*from*] [*to*]

Parameters

soundfile – handles a variety of standard and multi-channel soundfiles.
NB: You need to supply the extension explicitly with this program.
from – optional start-time in seconds (Default = 0).
to – optional end-time in seconds (Default= end-of-file)
Enables arbitrary block to be played and looped.

-d*N* – use output Device *N*
-g*N* – apply gain factor *N* to input
-B*N* – set memory buffer size to N frames (default: 32768)
N must be a power of 2 (e.g 4096, 8192, 16384 etc).
-h*N* – set hardware blocksize to N frames (32 < N <= BN/4). Default: from soundcard.
N is recommended to be a power of two size.
Where set, buffer sizes are doubled internally for sample rates >48KHz.
-i – play immediately (do not wait for keypress)
-l – loop file continuously, from start-time to end-time
-m[*S*] – render using channel map string S. (Use -m without parameter for usage.)
S (order string) = any combination of characters a-z inclusive.
No. of characters in S sets no. of output channels – must be supported by selected device.
Further details – see **CHORDER** *orderstring*
NB: -m cannot be combined with -b (B-Format decoding).
If channel mapping is used, a B-format file (AMB) will be rendered without decoding.
-u – suppress elapsed time updates
-x – apply WAVE_EX infile channel mask to Direct Sound audio stream (ignored if -m or -b used)
-b*N* – apply 1st-order B-Format decoding to a standard soundfile, which must have at least 3 channels.
B-Format (**.amb**) files will be decoded automatically.
The output channel order is anticlockwise from Centre Front except where indicated.
Layouts marked with an asterisk (*) are compatible with WAVE_EX speaker position order.
***N* sets the 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 (WAVE_EX order)
7. – * 5.1 surround (WAVE_EX order, silent LFE)
8. – hexagon
9. – octagon 1 (front pair, 45°)
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) NB: no decoding is performed if the -m flag isused.

Please also note:

- On Windows, ASIO as well as DirectSound is supported.
- OS X version supports Jack as well as Core Audio.
- Windows only: use the **-x** flag option to send WAVE_EX speaker positions (for starred layouts) to the hardware. The default is to send a zero mask, so that channels map directly to successive hardware outputs.
- PAPLAY reads the PEAK chunk if present to rescale over-range floating-point ('floatsam') files.
- Type PAPLAY without arguments to display 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 (see example below).
- For ASIO multi-channel, you may need to select the highest device number, for example:

Device	Input	Output	Name
0	2	0	Primary Sound Capture Driver
1	2	0	Microphone (Realtek High Definition Audio)
2	2	0	Monitor (M-Audio Delta Audiophile)
3	2	0	Line 1/2 (M-Audio Delta Audiophile)
4	2	0	S/PDIF (M-Audio Delta Audiophile)
*5	0	2	Primary Sound Driver
6	0	2	Line 1/2 (M-Audio Delta Audiophile)
7	0	6	Speakers (Realtek High Definition Audio)
8	0	2	S/PDIF (M-Audio Delta Audiophile)
9	0	2	Realtek Digital Output (Realtek High Definition Audio)
10	6	4	M-Audio Delta ASIO

About PAPLAY

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

The **-i** flag is provided primarily to enable the program to be invoked from a scripting language (e.g., TCL/Tk, MatLab, Octave, Tabula Vigilans, etc.) or from batch files, 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 MULTI-CHANNEL 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 **Advanced Mixing Options** enabled, to save each track as a channel of a multi-channel soundfile. 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 AIFFC 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 MULTI-CHANNEL TOOLKIT.

Windows
PAPLAY supports both the DirectSound and the ASIO APIs. Accordingly, the displayed list of devices will be longer than before, with the same hardware identified for each API as appropriate. In Windows, it is not necessary to set a device number if you wish to play the file on the default audio playback device set in Control Panel.

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 these 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, such as being able to audition an arbitrary multi-channel file over stereo speakers. Many commercial editors and DAWs perform in much the same way, sending the WAVE_EX channel mask directly to the subsystem. Should the user want to replicate this behaviour 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 soundfile 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 or octagon, this automatic remapping is not useful beyound 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.

End of PAPLAY

PUTCOL – place a column of numbers into a textfile

Usage

putcol *columnfile intofile out_textfile colno* **-r**|**-i** [[-]*skiplines*] [**-e**]

Parameters

columnfile – input textfile containing the column to be placed
intofile – name of existing textfile into which the column will be placed
out_textfile – name of resultant textfile
colno – position number of the column to be added (count starts at 1)
-r – replace original column
-i – insert new column amongst the original columns
skiplines – number of lines to skip from head of *intofile* before writing the coluimn data, retaining the lines skipped over (each blank line counts as one line)
-skiplines – number of lines to skip from head of *intofile* before writing the coluimn data, discarding the lines skipped over (each blank line counts as one line)
-e – Ignore lines which contain only 'e' or 's' (possibly legacy values?) and comment lines starting with ';'. (Skipped lines should still be counted).

Understanding the PUTCOL Function

PUTCOL saves typing time by placing a whole column of numbers into a textfile already containing at least one column of numbers. For example, if you have extracted and processed numbers extracted from a breakpoint file, you can use PUTCOL to replace the original column.

Note that, as with all CDP functions, the original file (*intofile*) is not actually over-written; instead you are writing to a new file (*out_textfile*).

Musical Applications

The COLUMNS program provides many ways of manipulating data, usually as single columns. For example, you might use it to quantise MIDI pitches. PUTCOL can place the processed column into a revised version of the original file.

ALSO SEE: GETCOL, COLUMNS, VECTORS

End of PUTCOL

PVPLAY – Play multi-channel PCM and analysis files

Usage

pvplay [**-B***N*] [**-d***N*] [**-g***N*] [**-h***N*] [**-i**] [**-l**] [**-b**[*N*] [**-u**] [**-x**] *infile* [*from*] [*to*]

Parameters

(identical to PAPLAY, except no -m flag)

infile [*.wav* | *.aif* | *.ana* | *.pvx*] – analysis file or soundfile to be played. **NB:** You need to supply the extension explicitly, so that the program knows what kind of file is being played.
from – start time (secs) for playback and looping. (Default: 0)
. *to* – end time (secs) for playback and looping. (Default: end-of-file)
-d*N* – use output Device *N*
-g*N* – apply gain factor *N* to input
-B*N* – set memory buffer size to N frames (default: 32768)
N must be a power of 2 (e.g 4096, 8192, 16384 etc).
-h*N* – set hardware blocksize to N frames (32 < N <= BN/4). Default: set internally.
N is recommended to be a power of two size.
Where set, buffer sizes are doubled internally for sample rates >48KHz.
-i – play immediately (do not wait for keypress)
-l – loop file continuously, from start-time to end-time
-u – suppress elapsed time updates
-x – apply WAVE_EX infile channel mask to Direct Sound audio stream (ignored if -b used)
-b*N* – apply 1st-order B-Format decoding to a standard soundfile, which must have at least 3 channels.
B-Format (**.amb**) files will be decoded automatically.
The output channel order is anticlockwise from Centre Front except where indicated.
Layouts marked with an asterisk (*) are compatible with WAVE_EX speaker position order.
***N* sets the 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 (WAVE_EX order)
7. – * 5.1 surround (WAVE_EX order, silent LFE)
8. – hexagon
9. – octagon 1 (front pair, 45°)
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)

Please also note:

- On Windows, ASIO as well as DirectSound is supported.
- OS X version supports Jack as well as Core Audio.
- Windows only: use the **-x** flag option to send WAVE_EX speaker positions (for starred layouts) to the hardware. The default is to send a zero mask, so that channels map directly to successive hardware outputs.
- PVPLAY reads the PEAK chunk if present to rescale over-range floating-point ('floatsam') files.
- Type PVPLAY without arguments to display 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 (see example below).
- For ASIO multi-channel, you may need to select the highest device number, for example:

Device	Input	Output	Name
0	2	0	Primary Sound Capture Driver
1	2	0	Microphone (Realtek High Definition Audio)
2	2	0	Monitor (M-Audio Delta Audiophile)
3	2	0	Line 1/2 (M-Audio Delta Audiophile)
4	2	0	S/PDIF (M-Audio Delta Audiophile)
*5	0	2	Primary Sound Driver
6	0	2	Line 1/2 (M-Audio Delta Audiophile)
7	0	6	Speakers (Realtek High Definition Audio)
8	0	2	S/PDIF (M-Audio Delta Audiophile)
9	0	2	Realtek Digital Output (Realtek High Definition Audio)
10	6	4	M-Audio Delta ASIO

Understanding the PVPLAY Function

PVPLAY enables you to audition spectral analysis files (with the extension **.ana**) – as well as soundfiles – through the device selected. The resynthesis is handled by PVPLAY in real-time.

PVPLAY makes it possible to audition the result of a spectral data manipulation process right away, prior to converting to a soundfile. This saves both time and disk space: you can hear and delete trial runs until achieving the desired result.

Because PVPLAY handles playback of both soundfiles and analysis files, it can be used as the default CDP play program. The Windows wave-mapper is useful as a way of finding a way to play back soundfiles which may not be supported by your main playback device.

N.B. While PVPLAY can play multi-channel spectral analysis files, identified by the **.pvx** extension (see usage), these have not yet been implemented in the rest of the CDP system.

PVPLAY in *Sound Loom*

When set in *Sound Loom* as the default play program, the **-i** flag should be set. The file which contains the information for playback for *Sound Loom* is *playcmd.cdp*, which is placed in the '_cdpenv' directory. If PVPLAY were to be set as the default play program, *playcmd.cdp* should contain:

```
pvplay -i
```

The *playcmd.cdp* file is an editable text file, and you should check that it contains what you expect it to contain. If for example, you set it from within *Sound Loom* itself, the **-i** flag (for immediate playback without user intervention) may not have 'taken'. You can put the flag there by editing the file.

End of PVPLAY

RECSF – Record, creating a soundfile (PC only)

Usage

recsf [**-B***N*] [**-c***N*] [**-d***N*] [**-h***N*] [**-i**] [**-p**] [**-r***N*] [**-t***N*] *outfile* [*dur*]

Parameters

outfile – output file in WAVE, AIFF or AIFC formats, determined by the file extension.
Use the extension **.amb** to create an ambisonic B-Format file.
-B*N* – set memory buffer size to N frames (default: 32768)
N must be a power of 2 (e.g 4096, 8192, 16384 etc).
-c*N* – number of channels to record (Range: 1 to 16; default = 2)
-d*N* – use input device *N*
Type **recsf** with no parameters for a list of available input devices.
Default device is marked with an asterisk (*). (See examples in **PAPLAY** and **PVPLAY**.)
For ASIO multi-channel, you may need to select the highest device number.
-h*N* – set hardware blocksize to N frames (32 < N <= BN/4). Default: set by device.
N is recommended to be a power of two size.
-i – start recording immediately (default: wait for keypress)
-p – suppress running peak level indicator
-r*N* – set sample rate to N Hz (default: 44100)
-t*N* – set sample type to N (default: 1)

- 0: 16 bit integer
- 1: 24 bit integer
- 2: 32 bit floating-point

dur – optional fixed duration (secs) for the outfile (overriden by Ctrl-C).
Otherwise, use Ctrl_C to terminate recording.

Understanding the RECSF Process

RECSF inputs the sound from the device selected, creating a soundfile in the specified format. It is best to start the playback device just before running RECSF so that any impulse generated when the device begins is not recorded.

The program also interrogates your system and displays the available recording time in seconds for a 16-bit soundfile at a sample rate of 44100.

End of RECSF

VECTORS – numerical operations between two columns of figures

Usage

vectors -h
OR
vectors *column1 column2* [*outfile*] *flag* [**-e***error*]

Parameters

-h – on-screen help message listing the flags
column1 – textfile with a list of numbers (on same or separate lines)
column2 – second textfile with a list of numbers (on same or separate lines)
outfile – text file to which the output is written. If not specified, output is to the screen.
flag – specifies the operation to perform. It can be one of:

-a – add each corresponding value of *column1* to *column2*
-s – subtract each corresponding value of *column1* from *column2*
-m – multiply each value of *column1* by the corresponding value of *column2*
-d – divide each value of *column1* by the corresponding value of *column2*
-p – raise each value of *column1* to the power expressed by the corresponding value of *column2* (must be >=0)
-Ra – add a random value between + and - the corresponding value of *column2*
-Rm – multiply *column1* value by a random value between 0 and the value in *column2*
-Rs – apply random variation to (ascending) values in *column1* (cumulatively) by adding it to a random value, the range of which is defined by the corresponding number in *column2* (which must be between 0 and 1); see further details below
-S*N* – substitute value *N* in *column1* with the next value in *column2*
-q*N* – quantise *column1* with *column2* (ascending order); see further details below
-o – overwrite first and every *N*th term of *column1* with the corresponding value of *column2*
-i*N* – insert *column2* at start and after every *N*th term of *column1*
-I – interleave *column2* alternately with *column1*

Understanding the VECTORS Function

Along with COLUMNS, VECTORS provides another range of facilities to manipulate columns of numbers.
The output is a single column, in effect a massaged *column1*; thus *column2* is being used to modify *column1*. This can be useful to 'loosen up' the use of numbers, to make them more 'plastic' and 'human'.

-Rs flag (random variation): in effect, this is 'humanising' the *column1* numbers with a random factor. *Column2* defines the range – always between 0 and 1 – within which the number is selected. For example, a 1 in *column2* means that the full 0-1 range will be used, whereas 0.5 in *column2* means that only half the range will be used, i.e., the random value will lie between 0 and 0.5.

-q flag (quantise): the values in *column1* are in effect 'snapped' to the nearest available value in *column2*.
For example:

column1	**column2**	**outfile**	Comment
2.5	3	3	
4.5	4	4	
6.5	9	4	6.5 is closer to 4 than to 9
8.5	10	9	8.5 is closer to 9 than to 10
10.5	12	10	10.5 is closer to 12 than to 10
12.5	15	12	12.5 is closer to 12 than to 15
	16	no output	outfile is truncated
	18	no output	

ALSO SEE: COLUMNS, **GETCOL**, **PUTCOL**.

End of VECTORS