NAME

libsoxeffect_ng – SoX effect handlers' internals

SYNOPSIS

```
#include "sox_i.h"

typedef struct {
    ...
} priv_t;

static int getopts(sox_effect_r * effp, int argc, char **argv);
static int start(sox_effect_r * effp);
static int flow(sox_effect_t * effp, sox_sample_t * ibuf,
                sox_sample_t * obuf, size_t * isamp, size_t * osamp);
static int drain(sox_effect_t * effp, sox_sample_t *obuf, size_t *osamp);
static int stop(sox_effect_t * effp);
static int kill(sox_effect_t * effp);
static char *get(sox_effect_t * effp, char *name);
static char *set(sox_effect_t * effp, char *name, char *value);

sox_effect_handler_t * effect(void)
{
  static char const usage[] = "[options]";
  static char const * const extra_usage = {
    "...",
    NULL
  };
  int const flags = 0;
  static sox_effect_handler_t handler = {
    "effect", usage, extra_usage, flags,
    getopt, start, flow, drain, stop, kill,
    sizeof(priv_t), get, set
  }
  return &handler;
}
```

DESCRIPTION

How SoX effects work internally and how to write a new one.

SoX's formats and effects operate with an internal sample format of signed 32-bit integers. The data processing routines are called with buffers of these samples and buffer sizes which refer to the number of samples processed. File readers translate input samples to signed 32-bit integers and return the number of samples read. For example, data in linear signed byte format is left-shifted 24 bits.

For effects that handle multiple channels simultaneously (see **SOX_EFF_MCHAN**), stereo data is stored with the left and right channels' data in successive samples and quadraphonic data is stored left front, right front, left rear, right rear and the number of samples that are available or have been processed is a count of mono samples, not of multichannel sample frames.

Each effect runs with one input and one output stream and its implementation comprises six principal functions that are called according to the following pseudocode:

```
LOOP (invocations with different parameters)
  getopts
  LOOP (invocations with the same parameters)
    LOOP (channels)
      start
```

```
  LOOP (while there is input audio to process)
    LOOP (channels)
      flow
  LOOP (while there is output audio to generate)
    LOOP (channels)
      drain
  LOOP (channels)
    stop
kill
```

Any function that an effect does not need can be NULL and any missing methods are automatically set to the appropriate **nothing** method.

getopts is called once when the effect is created, with an argv-like array of string arguments where **argv[0]** is the effect's name and **argv[1]** onward contain the effect's options as strings, the same as on the SoX command line, except for input and output where **argv[1]** is a pointer to the **sox_format_t** they should read or write.

start is called with the signal parameters for the input and output streams.

flow is called with input and output data buffers, and (by reference) the input and output data buffer sizes. It processes the input buffer into the output buffer, and sets the size variables to the numbers of samples actually processed: the number read and the number written. It is under no obligation to read from the input buffer or write to the output buffer during the same call. If the call returns **SOX_SUCCESS** it will we calle again; if it returns **SOX_EOF**, this means that the effect will not read any more data and can be switched to drain mode.

drain is called when there are no more input data samples. If the effect wishes to generate more data samples, it copies the generated data into the given buffer and writes the number of samples generated into ***osamp**. If it returns **SOX_SUCCESS**, it will be called again; **SOX_EOF** means it has no more samples to write.

stop is called when there are no more input samples and no more output samples to process. It is typically used to close or free resources such as memory and temporary files that were allocated during *start*.

kill is called to allow resources allocated by *getopts* to be released.

get is used by the **keymap** mechanism to read the value of one of the effect's parameters from its **priv_t**, returned as a string formatted with **sprintf** in `"%g"` format in mallocked memory that it is the caller's responsibility to free, or NULL if there is no such readable parameter.

set is used to set a parameter's value. The *name* of the parameter should match what it is called in the effect's **usage** string but with underscores instead of hyphens. and numerical values are passed as numeric strings (**sprintf(3)**'s `%g` format is recommended).

It returns a mallocked string version of the value that was actually set and which it is the caller's rsponsibility to free. The return values are sprintfed with `"%g"` so, if the caller does the same it can **strcmp()** the strings to see it it was set to the requested value or something different, which happens if the specified value is outside the parameter's range, in which case the maximum or minimum is set. It returns NULL is there is not a settable field of that name or if the value is garbage.

Effects that can have multiple stages, like **echo** and **chorus**, accept field names like **decay2** to adjust only the **decay** of the second stage; otherwise the parameter is changed in all of its stages.

FLAGS

The **flags** field tell SoX more about how the effect behaves. It is the logical OR of a number of bits:

SOX_EFF_CHAN
The effect might alter the number of channels.

SOX_EFF_RATE
The effect might alter the sample rate.

SOX_EFF_PREC
The effect does its own calculation of output sample precision; otherwise a default value is taken, depending on the presence of SOX_EFF_MODIFY.

SOX_EFF_LENGTH
The effect might alter the length of the audio as measured in time units, not necessarily in samples.

SOX_EFF_MCHAN
If **SOX_EFF_MCHAN** is not included in an effect's **flags**, the middle four functions are called once per channel with mono data, the channels may be processed in parallel, each channel has its own copy of the priv_t **that was filled in by start** and **effp->flow** tells them which channel they are working on (0 and 1 in the case of stereo).

If **SOX_EFF_MCHAN** is included, the effect does not use the LOOP (channels) lines and its **start**, **flow**, **drain** and **stop** functions are called once to process multichannel data with the samples interleaved, there is only one copy of its **priv_t** and **effp->flow** is always zero.

SOX_EFF_NULL
The effect does nothing. The **start** function can return this value to say it can be optimized out of the chain. This is done, for example, by **vol 1** and trim 0**.**

SOX_EFF_GAIN
The effect does not support the **gain −h** ... **gain −r** mechanism whereby **gain −h** automatically provides headroom for effects between itself and the **gain −r** or the end of the effects chain.

This works through the **mult** field of **sox_effect_t** which is set to point to the volume adjustment of the **gain −h** effect and is copied forwards from each effect to the next so that the **gain −r** can

Effects that know the maximum gain they could apply, at the end of their **start** function, divide ***effp->in_signal.mult** by whatever their linear amplification is or if, for example, they know that their maximum output has half the amplitude of their maximum input, they multiply it by two. If there is no previous **gain −h** in the effects chain, **mult** is NULL.

Effects that do not affect the maximum signal amplitude do not include **SOX_EFF_GAIN** in their flags and need do nothing else; the pointer to **gain −h**'s volume adjustment is copied forwards in the chain for them.

Confusingly, **SOX_EFF_GAIN** means "I *don't* support what **−h** and **−r** need because I don't know how much I might change the maximum amplitude by and I neither copy the pointer forwards nor modify what it points to"; such an effect breaks the link between **gain −h** and **gain −r** and a warning will be issued to advise of this.

SOX_EFF_MODIFY
The effect does not modify sample values but might remove or duplicate samples or insert zeros.

SOX_EFF_INTERNAL
The effect is present in libSoX but is not valid for use by the SoX command-line tools. It applies to **input** and **output**.

SEE ALSO

sox_ng(1), **libsox_ng**(3), **soxeffect_ng**(7)**and** `src/skeleff.c` in the SoX source code.