Getting started with Clive

Claude Heiland-Allen

2018-09-03

Getting started with Clive

Dependencies

Debian

```
# apt install \
    sudo git ca-certificates \
    build-essential pkg-config \
    libjack-jackd2-dev qjackctl \
    cpufrequtils ecasound \
    xterm htop geany \
    python-pygments
```

Configuring `sudo` for `cpufrequtils`

Replace `claude` with your username and `latte` with your machine name:

```
$ sudo visudo /etc/sudoers.d/cpufreq-set
claude latte = (root) NOPASSWD: \
  /usr/bin/cpufreq-set
```

The file contents should be one line without \, just split to fit slides!
This allows to change CPU frequency governor without password.

Configuring JACK for realtime

Provided by JACK packaging on Debian:

```
$ cat /etc/security/limits.d/audio.conf
@audio   -  rtprio     95
@audio   -  memlock    unlimited
```

To check that you are in the audio group:

```
$ groups
```

Download `clive` repository

```
git clone https://code.mathr.co.uk/clive-core.git
```

Launching

Launch `qjackctl` and configure your sound card.
Launch `clive`:

```
cd clive-core/src
./start.sh
```

After a short delay, there should be 2 terminal windows on the left and the `geany` text editor which can be resized to fit on the right.
Edit `go.c`, press Ctrl-S to save, which will recompile and reload the code.

Exiting

- Ctrl-C in the terminal running `./start.sh`
- Stop and rewind JACK transport
- `git checkout master`

Example: a metronome

```
git checkout origin/metronome
git checkout -b metronome
```

Example: a metronome

```
#define SR 48000
#include "dsp.h"

typedef struct {
  int reloaded;
  PHASOR clock, osc;
} S;

int go(S *s, int channels, const float *in, float *out) {
  if (s->reloaded) { s->reloaded = 0; }
  double env = phasor(&s->clock, 125/60.0) < 0.25;
  double osc = sin(twopi * phasor(&s->osc, 440));
  double o = env * osc;
  for (int c = 0; c < channels; ++c) { out[c] = o; }
  return 0;
}
```

Clive links

- claude@mathr.co.uk
- https://mathr.co.uk/clive