Benben v0.7.0

Remilia Scarlet

Short Contents

Table of Contents

1 Overview of Benben

1.1 Introduction

Benben is a fast and efficient command line audio player and audio converter for Linux and other Unix-like systems with an oldschool-inspired interface. It supports multiple formats, and is especially suited to people who organize their music in folders, and for those who prefer to use terminals instead of GUIs.

Benben is written almost entirely in the Common Lisp programming language.

1.2 Supported Input Formats

- VGM files (`https://vgmrips.net/`)
  - Uncompressed (.vgm)
  - GZip compressed (.vgz)
  - ZStandard compressed (.vgzst)
  - BZip2 compressed (.vgb)
- MPEG-1
  - MPEG-1 Layer III (.mp3)
  - MPEG-1 Layer II (.mp2)
  - MPEG-1 Layer I (.mp1)
- Ogg Vorbis
- Opus
- FLAC
- General MIDI (.mid, .rmi)
- MUS (.mus)
- All module formats supported by libxmp (`https://github.com/libxmp/libxmp/`)
- Commodore 64 SID (.sid)
- RIFF WAVE (.wav)
- QOA (`https://qoaformat.org/`), the Quite OK Audio format
- XQAF (`https://chiselapp.com/user/MistressRemilia/repository/cl-remiaudio/file?name=docs/extended-qoa-format.md&ci=tip`), the Extended Quite OK Audio Format
- WavPack
- Sun Au (.au)

In addition to the above formats, Benben can also read songs contained within Doom WAD files (`https://doomwiki.org/wiki/WAD`). All formats except SID can be loaded from WADs.

1.3 Supported Output Formats

- RIFF WAVE (.wav), PCM or IEEE float samples
- Sun Au (.au), PCM or IEEE float samples

1.4 History

Before Benben ever existed, Remilia started work on a port of a SoundFont synthesis library called MeltySynth[1] from C# to Crystal. Her port of this software was called[2]. As part of her port, she included a simple command line player called midi123, which was very bare bones and was intended to be nothing more than an example/test player for the library. However, it quickly grew to be more than a simple example and eventually became a standalone command-line player for General MIDI files.

About a year later, Remilia started to work on creating a VGM playback library called YunoSynth[3] for native Crystal. This was partially a port of vgmplay[4] from C, and partially an object-oriented rewrite. Similar to midi123, and using its code as a base, Remilia also created a companion command-line player for YunoSynth called Benben.

The first few version of Benben only played VGM files (or their compressed variants), and used raw ANSI console commands to draw its interface. Starting with v0.3.0, Benben moved to an interface built with S-Lang (more accurately, her own Crystal bindings to S-Lang). Remilia also found herself continually wishing that Benben could play other formats so that she wouldn't have to switch between audio players, and so beginning with v0.5.0, Benben became a general-purpose music player that supports multiple input formats.

Benben (and all of its support libraries) underwent a complete rewrite for v0.7.0 in order to be ported to the Common Lisp programming language.

[1] `https://github.com/sinshu/meltysynth/`

[2] Haematite (`https://chiselapp.com/user/MistressRemilia/repository/Haematite/`)

[3] `https://chiselapp.com/user/MistressRemilia/repository/yunosynth/`

[4] `https://github.com/vgmrips/vgmplay/`

2 Basic Usage

2.1 Playing Files

Benben is a command line program. To play a file, simply pass it as an argument to Benben:

```
benben coolsong.mp3
```

Multiple files can also be specified, one after another. They will be played in the order that you list them:

```
benben coolsong1.mp3 other-directory/coolsong2.flac coolsong3.vgm
```

If you specify a directory, then Benben will play all files in that directory in alphabetical order. It will not recurse into any subdirectories by default, however.

```
benben directory-with-music/
```

To recurse subdirectories, add the `--recurse` option. You can also specify this as the default in your configuration file if you'd like (see [recurse-subdirectories], page 17).

```
benben --recurse directory-with-music/
```

Playlists in XSPF[1] or JSPF[2] format can also be used. These can be passed in like any other file, and their contents will be queued up in the same order that the playlist specifies.

```
benben cool-mix.xspf
```

Finally, you can mix and match any of these methods as you see fit:

```
benben cool-mix.xspf coolsong1.flac /directory/with/music/
```

2.2 User Interfaces

Benben comes with two user interfaces: the "Original UI" and the "Minimal UI". The Original UI is the default and, prior to Benben v0.7.0, was the only user interface available. When in doubt, stick with the Original UI.

2.2.1 The Original User Interface

During playback, a progress bar will be displayed at the bottom. It will look something like this:

```
[EsCr-] 1/1, 1 of 2: |*****************--------------------|  45% [02:20/05:09]
```

The characters to the left in brackets indicate the state of various effects and other pieces of information. These are, in order from left to right:

'`E`' EQ is on.

'`e`' EQ is off.

'`S`' Stereo enhancer is on.

'`s`' Stereo enhancer is off.

'`C`' Soft clipping is on.

[1] `https://xspf.org/`

[2] `https://xspf.org/jspf`

'`c`' Soft clipping is off.

'`R`' Reverb is on.

'`r`' Reverb is off.

'`+`' A song-specific config was found for the current song and loaded. See Section 5.3 [Song-specific Config Files], page 29,

'`-`' A song-specific config was not found for the current song. See Section 5.3 [Song-specific Config Files], page 29,

Next to these characters will be the current track, a slash, and then total number of tracks. Following this is the loop information (e.g. '`1 of 2`' means "currently in the first loop out of two"). When the loop information displays as '`1 of *`', then the song is looping indefinitely. Note that for VGM files, if the song does not have any loop information in it, then this will always read as '`1 of 1`'. The loop information will be changed to the text "fading" when a song is being faded out by Benben.

There is some additional information located at the bottom of the user interface. This will look something like this:

```
v0.7.0-rc3, Press 'q' to quit, 'h' for help      [rsp] CPU: 0.0%      Vol: 1.00
```

Starting from the left is the version of the program, a message indicating how to display basic help information at runtime, global state information (see below), CPU usage for Benben itself, and the output volume of the program.

The global state information (the `[rsp]` field) indicates the state of various global settings for the entire program. These are, in order from left to right:

'`R`' The entire song queue will be repeated when it reaches the end.

'`r`' The entire song queue will not be repeated when it reaches the end.

'`S`' The program will exit after the current song finishes playing ("stop after current" mode is enabled).

'`s`' The program will continue as normal after the current song finishes playing ("stop after current" mode is disabled).

'`P`' The next song will automatically start in a paused state after the current song finishes playing ("pause after current" mode is enabled).

'`p`' The next song will start normally after the current song finishes playing ("pause after current" mode is disabled).

2.2.1.1 Keyboard Input

The program will respond to various keys during playback. These are listed below, and you can also press the *h* key to see them while Benben is running.

n Go to the next file in the playback queue. If the `--repeat` option is used and you are already on the last item in the queue, then Benben will loop back to the first item. See Chapter 3 [Command Line Options], page 7,

p Go to the previous file in the playback queue. If you are already on the first item, Benben will simply restart the playback of that item.

`t`	Toggle the displayed GD3 tag language. Only works for VGM files.
`i`	Toggle the interpolation mode. Only for module/tracker formats.
`c`	Toggle soft clipping on and off.
`e`	Toggle the EQ effect on and off.
`d`	Toggle the DC offset filter on and off.
`s`	Toggle the stereo enhancer effect on and off.
`r`	Toggle the reverb effect on and off.
`a`	Raise the volume.
`z`	Lower the volume.
`]`	Increase the number of times the song loops. Not all formats support this.
`[`	Increase the number of times the song loops. Not all formats support this.
`q`	Quit the program.
`R`	Redraws the entire screen.
`S`	Toggles the "stop after current song" mode. When this is enabled, Benben will exit once the current song finishes playing or you hit `n` to move to the next file.
`U`	Toggles the "pause after current song" mode. When this is enabled, then the next song will be start in a paused state after the current song finishes playing or you hit `n` to move to the next file.
`C`	Reloads all song-specific configuration files. See Section 5.3 [Song-specific Config Files], page 29,
`T`	Reloads the current theme from disk.
`>`	Seeks forward a bit. Not all formats support this.
`<`	Seeks backwards a bit. Not all formats support this.
`x`	Toggles the chorus effect. This only works for General MIDI files.
`I`	Show extended information about the currently applied effects and resampler.
`space`	Pause/unpause playback.
`P`	Toggles repeat of the entire song queue. In other words, when this is enabled, then Benben will loop back to the very first song in the song queue once the final song finishes playing. When this is disabled, then Benben will exit after the final song finishes playing.

2.2.2 The Minimal User Interface

Starting with Benben v0.7.0, an alternate user interface called the "Minimal UI" exists that gives power users an alternate way to experience Benben. As its name implies, this interface is very minimal, to the point that it does not even have key bindings. All input is instead handled through the `remote-benben` program (see Chapter 7 [Remote Control], page 37). No ANSI sequences are used, and only plain text is printed to the terminal.

This user interface is most useful for users who have a headless setup, who want to control Benben entirely remotely and do not need a full UI, and for debugging Benben. Starting the Minimal UI is accomplished by passing `--ui minimal` in as a command line argument.

The typical output of the program in this mode starts with the usual startup text, followed by the "Benben" banner. Basic output information is printed after this, such as the output sample rate, the number of tracks, the total play time, and so on. After this, the information that gets displayed depends on what sort of events occur. For example, track information will be displayed each time Benben plays a new track. Likewise, changing any of the effect settings will cause effect information to be displayed.

```
Output Sample Rate:  44100
Resampler:           sinc-best
Backend:             out123
Total files to play: 1
Total play time:     02:24

Effects:
  EQ:              on, Reverb: off
  DC Filter:       on, Soft Clipping: on (1x)
  Stereo Enhancer: off

Effects changed:
  EQ:              off, Reverb: off
  DC Filter:       on, Soft Clipping: on (1x)
  Stereo Enhancer: off

Track changed:
  Track:       1/1
  Type:        MPEG-1
  Info:        MPEG-1 Layer III File, 320 kbps CBR
  Filename:    01 - Police Truck.mp3
  Title:       Police Truck
  Artist:      Dead Kennedys
  Album:       Give Me Convenience or Give Me Death
  Date:        1987
  Genre:       Punk
  Sample Rate: 44100
  Length:      02:24
```

3 Command Line Options

General Options

`--help, -h`
: Displays help information, then exits.

`--version, -v`
: Displays brief version information, then exits.

`--long-version`
: Show verbose version information, then exits.

`--scan-only`
: Only scan files, then exit.

`--recurse`
: Recursively look for files in sub-directories. Overrides the `recurse-subdirectories` configuration option (see [recurse-subdirectories], page 17).

`--no-recurse`
: Do not recursively look for files in sub-directories. Overrides the `recurse-subdirectories` configuration option (see [recurse-subdirectories], page 17).

`--dump-config`
: Generates a fresh configuration file and prints it to standard output, then exits.

`--basedir` *x*
: Load configuration data and themes from an alternate directory.

`--remote`
: Start up a UNIX domain socket for remote control. See Chapter 7 [Remote Control], page 37,

`--remote-socket` *x*
: Put the socket for the `--remote` option in an alternate location. See Chapter 7 [Remote Control], page 37,

`--list-themes`
: Lists all available themes. See Chapter 6 [Themes], page 32,

`--theme` *x*`, -T` *x*
: Use the specified theme. See Section 6.1 [Specifying a Theme], page 32,

`--ui` *x*
: Use a specific user interface. Use `--ui list` to see a list of available user interfaces. The default is `original`.

`--no-listenbrainz, -L`
: Never submit to ListenBrainz.

TCP Driver Options

`--tcp-driver-host` *x*
: The host connect to when using the TCP audio driver. The default is `localhost`. See Chapter 8 [Sending Audio Over TCP], page 41,

`--tcp-driver-port` *x*
: The port connect to when using the TCP audio driver. The default is `6969`. See Chapter 8 [Sending Audio Over TCP], page 41,

`--tcp-driver-format` *x*
: The audio format to transmit to when using the TCP audio driver. Use `--tcp-driver-format list` to see a list of available formats. The default is `float32`. See Chapter 8 [Sending Audio Over TCP], page 41,

Sound Options

`--driver` *x*`, -d` *x*
: Selects the audio driver to use. Use `--driver list` to see a list of available drivers. The default is `any`.

`--alsa-device` *x*
: Use a specific ALSA device; ignored if not using the ALSA driver. Default: `default`.

`--volume` *x*`, -v` *x*
: Sets the output volume. The valid range is 0.0 to 3.0, and the default is 1.0.

`--sample-rate` *x*`, -S` *x*
: Sets the output sample rate, in hertz. Valid range is 8000 to 48000, and the default is 44100.

`--replay-gain` *x*
: Set the ReplayGain mode. This only applies to files that support ReplayGain. Use `--replay-gain list` to see a list of values; The default is `disabled`.

`--resampler` *x*
: Sets the resampler mode. Use `--resampler list` to see a list of available modes. The default is `linear`.

`--no-eq, -e`
: Start with the equalizer disabled. The equalizer can still be toggled during playback with the *e* key.

`--soft-clipping, -c`
: Start with soft clipping enabled on the output. This can still be toggled during playback with the *c* key.

`--no-soft-clipping, -C`
: Start with soft clipping disabled on the output. This can still be toggled during playback with the *c* key.

`--soft-clipping-oversampling` *x*
: Changes the amount of oversampling that's applied when doing soft clipping. The default is `1`, which means no oversampling is performed by the soft clipper.

`--soft-clipping-mode` *x*
: Sets the type of resampler to use for oversampling when soft clipping is enabled (default: `linear`). Uses the same values as `--resampler`.

`--stereo-enhancer, -t`
: Start with the stereo enhancer effect enabled. This can still be toggled during playback with the ***s*** key.

`--stereo-enhancement` *x*`, -E` *x*
: Sets the stereo enhancement amount when the stereo enhancer is enabled. The valid range is 0.0 (nearly monaural) to 1.5 (very wide). The default is 0.5, which is equivalent to not having the effect enabled. If the stereo enhancer is not enabled, this does nothing.

`--dc-filter`
: Start with the DC offset filter enabled.

`--no-dc-filter`
: Start with the DC offset filter disabled.

`--reverb, -r`
: Start with the reverb effect enabled. This can still be toggled during playback with the ***r*** key.

`--no-reverb, -R`
: Start with the reverb effect disabled. This can still be toggled during playback with the ***r*** key.

`--reverb-type` *x*
: Set the type of reverb unit that is used. You can use `--reverb-type list` to see a listing of valid types. The default is `mverb`.

`--reverb-preset` *x*
: Changes the selected reverb unit's parameters to an alternate preset. You can use `--reverb-preset list` to see a listing of valid types for the selected reverb unit. The default is `gm-default`.

`--reverb-amount` *x*`, -r` *x*
: The amount of reverb to apply to the output. Valid range is 0.0 to 1.0 default is 0.5. This does nothing if the reverb is disabled.

Playback Options

`--loop` *x*`, -l` *x*
: The number of times to loop during playback and rendering; a value of 0 means to loop indefinitely. A value of 0 cannot be used when rendering to a file. Only certain formats support looping.

`--repeat, -p`
: Go back to the first song after the last one has finished playing.

`--no-repeat, -P`
: Do not go back to the first song after the last one has finished playing. This is mainly meant for overriding the config file setting.

`--shuffle`
: Randomizes the playback queue before starting playback.

`--start-paused`
: Start the program in a "paused" state.

`--fadeout-seconds` *x*
: How many seconds to fade out when Benben fades out a song. This only applies to certain formats. Default: 5.

VGM Options

`--vgm-chip-info`
: Displays a list of supported VGM chip emulators, then exits.

`--vgm-strict-gd3-loading`
: Be strict about GD3 tags when loading the VGM. This disables parallel file checking at startup.

`--vgm-dmg-boost-wave-chan`
: Boost the WAVE channel in the DMG emulator core.

`--vgm-no-dmg-boost-wave-chan`
: Do not boost the WAVE channel in the DMG emulator core.

`--vgm-ym2612-pseudo-stereo`
: Enable the pseudo-stereo effect for the YM2612 emulator core.

`--vgm-no-ym2612-pseudo-stereo`
: Disable the pseudo-stereo effect for the YM2612 emulator core.

Commodore 64 SID Options

`--sid-mono`
: Force 2SID and 3SID files to playback as monaural. See Chapter 4 [Playing SID Files], page 12,

`--sid-def-length` *x*
: Change the default length for SID songs, in seconds. This is useful for SIDs that are not found within a song length database, or when no song length database is used. The default is `180`. See Chapter 4 [Playing SID Files], page 12,

MIDI Options

`--midi-soundfont` *x*
: Use a specific SoundFont for all MIDI files.

`--midi-reverb-type` *x*
: Sets the type of reverb for MIDI files. This doesn't affect other formats. Default: `mverb`. Use `--midi-reverb-type list` to see valid types.

Rendering Options

`--render, -n`
: Tells Benben to render files to WAV (or Au) rather than play them back. Each input file is rendered to its own destination file, where the output name matches the original filename. This renders to WAV by default.

`--quiet, -Q`
: Don't print any messages or progress, except errors.

`--normalize, -N`
: Normalize each rendered file so that it's peak is zero dBFS.

`--cue` *x*
: Write a CUE file for the rendered files.

`--au, -A`
: Render to Au instead of WAV.

`--float, -F`
: Output files will contain IEEE Floating Point data instead of integer PCM data.

`--bit-depth` *x*`, -b` *x*
: Set the bit depth for the output files. Valid bit depths are 8, 16, 24, 32, and 64. The default is 16 when rendering integer PCM data, and 32 when rendering IEEE floating point data.

`--outdir` *x*`, -o` *x*
: Where to save the rendered files.

`--overwrite`
: Overwrite existing files when rendering.

`--jobs` *x*
: The number of parallel rendering jobs to use. This is only meaningful when rendering more than one input file. This must be at least 1, and the default is equal to the number of logical CPU cores.

4 Playing SID Files

Commodore 64 SID files are files that store music composed for the Commodore 64. Benben can play these by emulating part of a Commodore 64 and its sound chip and the MOS 6581 "SID" chip (or the MOS 8580). Benben handles these somewhat differently than other formats because a single SID file is capable of storing more than one song, and there are some configuration options that are specific to SID files. See Chapter 5 [Configuration], page 14,

The Commodore 64 SID chip was a monaural sound chip, so SID songs are not stereo by default. However, some songs support multiple SID chips, known as 2SID and 3SID (for two chips and three chips, respectively). Stereo output is possible with these files, and Benben has full support for these without needing to do anything special. You can force these to output as monaural using the `--sid-mono` command line option. See Chapter 3 [Command Line Options], page 7,

4.1 Playing SID Songs

To play all songs in a SID file, simply pass it in on the command line like usual:

```
benben CYBERNOID.SID
```

Selecting a single song out of a SID file is also possible, for example to select the second song in a SID:

```
benben LAST_NINJA_2.SID@2
```

you can also select a set of individual songs:

```
benben ROBOCOP3.SID@2,4,7
```

Finally, a range of songs can also be selected:

```
benben COMMANDO.SID@1-3
```

All of these techniques are also possible from playlist files.

4.2 SID Songlength Database

SID files are unusual in that they do not contain information about how *long* a song is. This is remedied by use of a *songlengths database*. This is an INI-format file that simply stores length information about known SID files. The High Voltage SID Collection (`https://www.hvsc.c64.org/downloads`) site has an up-to-date songlengths database file available for download that will cover nearly all known SID files, and is the recommended way to inform Benben about the length of SID songs.

To use a database, first download it and place it somewhere on your computer. Then open up your Benben configuration file and add this line, replacing the path with the path of the file you just downloaded. Note that you may already have a `c64` section; if you do, add the `song-length-db` line under your existing section, otherwise create a new `c64` section. See Chapter 5 [Configuration], page 14,

```
c64:
  song-length-db: /path/to/your/Songlengths.txt
```

If a song is not found in the database, or if you do not use a songlength database, then Benben will instead play the song using a "default" song length. This length is 180

seconds by default, but can be configured to a different length in your configuration file. See Chapter 5 [Configuration], page 14,

```
c64:
  default-song-length: 300  # This value is in seconds
```

You can also change this on command line using `--sid-def-length`. See Chapter 3 [Command Line Options], page 7,

4.3 ROM Files

Benben will play most SID files without issue out-of-the-box. However, a few SID files may require dumps of various ROM chips from a Commodore 64, specifically the `KERNAL`, `BASIC`, and `CHARGEN` ROMs. You should be able to find these with a bit of clever Internet searching. Once you have downloaded these, place the relevant options into Benben's configuration file. See Chapter 5 [Configuration], page 14,

```
c64:
  kernal-rom: /home/remilia/doc/c64/kernal
  basic-rom: /home/remilia/doc/c64/basic
  chargen-rom: /home/remilia/doc/c64/chargen
```

5 Configuration

Benben's configuration file lets you specify many of its options, saving you from having to type them in on the command line each time, as well as a few that aren't available via the command line. The file uses the YAML[1] format for ease of use. The first time you run Benben, a default configuration file will be created. If you wish to create a fresh one at a later date, you can run `benben --dump-config`, which will print a fresh configuration file with the default values to standard output.

The location of the configuration file depends on your platform. On Linux/Unix, the file is located at `$XDG_CONFIG_HOME/benben/benben.yaml` (this usually means `~/.config/benben/benben.yaml`).

Note that command line options can still override the configuration file. So you can have your configuration file set to loop each song 3 times, you still use the `--loop` parameter to temporarily change to another value. See Chapter 3 [Command Line Options], page 7,

Important Note

The configuration format is not considered stable, and won't be until Benben v1.0 is released. Until then, you *may* have to make small adjustments when upgrading between versions. These changes will be noted in the release notes for each version of Benben.

5.1 Example Configuration File

```
buffer-size: 256
audio-driver: out123
sample-rate: 44100
fadeout-seconds: 5
repeat-playlist: false
render-jobs: 8
main-volume: 1.0
max-loops: 1
resampler: sincbest
replay-gain: album
tcp-driver-format: f32
remote: true
recurse-directories: false

listenbrainz:
  token: 123-abc-your-token-here

enable-stereo-enhancer: false
stereo-enhancement-amount: 1.1
no-soft-clipping: false
soft-clipping-oversampling: 1
reverb-type: mverb
reverb-enabled: false
```

[1] https://en.wikipedia.org/wiki/YAML

```
reverb-amount: 0.369

vgm:
  preferred-gd3-lang: toggle_prefer_english

modules:
  default-panning: 69
  interpolation: spline
  fade-out-songs: false

midi:
  soundfont: "/home/remilia/doc/soundfonts/sc-55.sf2"
  reverb-enabled: true
  reverb-type: mverb

c64:
  song-length-db: /home/remilia/doc/c64-songlengths.txt
  default-song-length: 180
  kernal-rom: /home/remilia/.local/share/sidplayfp/kernal
  basic-rom: /home/remilia/.local/share/sidplayfp/basic
  chargen-rom: /home/remilia/.local/share/sidplayfp/chargen
  default-c64-model: ntsc
  default-sid-model: mos6581
  sampling-method: ResampleInterpolate

ui:
  animations-enabled: true
  theme:
    - plum
    - smooth-neon
    - blue-phosphor
    - phosphor
    - semi-retro
    - remilia
  banner: [graffiti, stronger, speed]
  display-loop-times-separate: true

equalizer-enabled: true
equalizer-disabled-during-rendering: true
equalizer:
  post-gain: 0.0
  low-shelf:
    frequency: 80.0
    gain: 2.0
    bandwidth: 1.0
  bands:
    - frequency: 105.0
```

```
      gain: 2.78
      bandwidth: 0.6
    - frequency: 275.0
      gain: -6.0
      bandwidth: 0.2
    - frequency: 6000.0
      gain: 1.5
      bandwidth: 1.0
  high-shelf:
    frequency: 9001.0
    gain: 1.4
    bandwidth: 0.9
```

5.2 Configuration Options

5.2.1 Main Options

Key Name	Type	Description
`audio-driver`	String	The output driver used to produce sound during playback. Use `benben --driver list` to see a list of valid options. Default: '`any`'. See Chapter 3 [Command Line Options], page 7,
`alsa-device`	One of: `out123`, `alsa`, `ao`, `alsa`, or `any`	The ALSA device to use when using the `alsa` driver; ignored when using any other driver. Default: '`default`'. See Chapter 3 [Command Line Options], page 7,
`buffer-size`	An integer that is at least 256, and evenly divisible by 256.	The size of the audio buffer. Note that this also affects the smoothness of the VU meter. In most cases, this doesn't need changing. Default: '`256`'.
`sample-rate`	An integer between 8000 and 48000.	The output sample rate, in hertz. Default: '`44100`'.

`tcp-driver-host`	String	The hostname of the machine to send audio to when using the TCP driver. This only affects the TCP `audio-driver`. Default: '`localhost`'. See Chapter 8 [Sending Audio Over TCP], page 41,
`tcp-driver-port`	An integer between 1 and 65535	The port of the machine to send audio to when using the TCP driver. This only affects the TCP `audio-driver`. Default: '`6969`'. See Chapter 8 [Sending Audio Over TCP], page 41,
`tcp-driver-format`	One of: `float64`, `float32`, `int64`, `int32`, `int24`, `int16`, `int12`, `int8`, or `uint8`	The raw audio format to send when using the TCP driver. This only affects the TCP `audio-driver`. Default: '`f32`', meaning 32-bit IEEE Floating Point. See Chapter 8 [Sending Audio Over TCP], page 41,
`recurse-directories`	`true` or `false`	When '`true`', then sub-directories are also searched when you pass Benben a directory. Default: '`false`'.
`repeat-playlist`	`true` or `false`	When '`true`', then the player will loop back to the first song once the final song is finished playing. This only has an effect during playback, not during rendering. Default: '`false`'.
`render-jobs`	An integer >= 1	The number of workers (threads) to use when rendering in parallel. This must be at least 1, and defaults to the number of logical CPU cores you have.
`fadeout-seconds`	An integer between 0 and 255	The number of seconds a looping song will fade out when finished. Default: '`5`'. This only applies to certain formats.

`no-soft-clipping`	`true` or `false`	When '`true`', soft clipping is disabled by default. Default: '`false`'.
`soft-clipping-oversampling`	An integer between 1 and 256	How much oversampling to apply when using the soft clipping effect. Default: '`1`', meaning "no oversampling".
`soft-clipping-mode`	One of: `sinc-best`, `sinc-medium`, `sinc-fast`, `zero-order-hold`, `linear`	The resampling mode to use for the soft clipper. This only takes effect when the `soft-clipping-oversampling` value is over '`1`'. Default: '`linear`'.
`enable-stereo-enhancer`	`true` or `false`	When '`true`', this enables a stereo enhancement effect. Default: '`false`'.
`stereo-enhancement-amount`	Float between 0.0 and 1.5	The amount of stereo enhancement to apply to the output signal. This requires `enable-stereo-enhancer` to be '`true`'. This can go from '`0.0`' (nearly monaural) to '`1.5`' (very wide). Default: '`0.5`', which is the same as not enabling this effect.
`main-volume`	Float between 0.0 and 2.0	The main output volume. Default: '`1.0`'.
`max-loops`	An integer between 0 and 4294967295	The maximum number of times to loop if the song has loop information. If the song has no loop information, this is ignored. A value of '`0`' means "loop forever". Default: '`1`' (meaning "play once, then loop once").
`resampler`	One of: `sinc-best`, `sinc-medium`, `sinc-fast`, `zero-order-hold`, `linear`	The resampler mode to use when playing a file that does not match the `sample-rate` setting. The default is fine in most cases. Default: '`linear`'.

`seek-time`	An integer between 0 and 65535	How far to seek when fast-forwarding/rewinding. The meaning of this value is format-dependent. Default: '`10`'.
`replay-gain`	One of: `disabled`, `mix`, or `album`	How to apply ReplayGain to formats that support it. Default: '`disabled`'.
`remote`	`true` or `false`	Whether or not to enable remote control via the `remote-benben` program at startup. Default: '`false`'. See Chapter 7 [Remote Control], page 37,
`dc-filter`	`true` or `false`	When '`true`', a DC offset filter will be applied to the output. You can still turn on and off the equalizer by pressing the *d* key during playback. Default: '`true`'.
`reverb-enabled`	`true` or `false`	When '`true`', then a reverb effect will be enabled by default. Default: '`false`'.
`reverb-type`	`mverb` or `zita`	The type of reverb to apply when reverb is enabled. Default: '`mverb`'.
`reverb-amount`	Float between 0.0 and 1.0	The amount of reverb to apply when reverb is enabled. Default: '`0.5`'.
`mverb-reverb-preset`	String	The preset to use for the reverb when reverb is enabled and `reverb-type` is set to '`mverb`'. Default: '`gm-default`'. Use the command line option `--reverb-preset list` together with `--reverb-type mverb` to see a list of presets for that type.

`zita-reverb-preset`	String	The preset to use for the reverb when reverb is enabled and `reverb-type` is set to '`zita`'. Default: '`gm-default`'. Use the command line option `--reverb-preset list` together with `--reverb-type zita` to see a list of presets for that type.
`equalizer-enabled`	`true` or `false`	When '`true`', an equalizer effect will be applied to the output. You can still turn on and off the equalizer by pressing the `e` key during playback. Default: '`false`'.
`equalizer-disabled-during-rendering`	`true` or `false`	When '`true`', no equalizer effect will be applied to the output when rendering a file to WAV/Au. This only takes effect when rendering to files. Default: '`false`'.
`equalizer`	Sub-block. See Section 5.2.8 [Equalizer Config Block], page 27,	Configuration section for the equalizer.
`listenbrainz`	Sub-block. See Section 5.2.9 [ListenBrainz Config Block], page 29,	The settings for ListenBrainz integration.
`ui`	Sub-block. See Section 5.2.4 [UI Config Block], page 21,	The settings for the user interface.
`vgm`	Sub-block. See Section 5.2.2 [VGM Config Block], page 21,	The settings for VGM files.
`modules`	Sub-block. See Section 5.2.5 [Modules Config Block], page 22,	The settings for module/tracker files.
`midi`	Sub-block. See Section 5.2.6 [MIDI Config Block], page 23,	The settings for MIDI/MUS files.
`c64`	Sub-block. See Section 5.2.7 [C64 Config Block], page 25,	The settings for C64 SID files. See Chapter 4 [Playing SID Files], page 12,

5.2.2 VGM Config Block

Key Name	Type	Description
‘preferred-gd3-lang‘	One of: `english`, `japanese`, `toggle-prefer-english` or `toggle-prefer-japanese`	The default language in which to display GD3 tag info. Default: ‘`japanese`’. This can be toggled during playback with the `t` key. Only applicable to VGM files.
`emulators`	Sub-block. See Section 5.2.3 [Emulator Config Block], page 21,	Sets emulator-specific settings for VGM files.

5.2.3 Emulator Config Block

Key Name	Type	Description
`dmg-boost-wave-chan`	`true` or `false`	Doubles the volume of the wave channel when playing GameBoy music. Default: ‘`true`’.
`huc6280-core`	`mame` or `ootake`	Determines the emulation core used for music that uses the HuC6280 chip (e.g. PC Engine/TurboGrafx-16). Default: ‘`ootake`’
`ym2151-core`	`mame`	Determines the emulation core used for music that uses the YM2151 chip. Default: ‘`mame`’.
`ym2612-pseudo-stereo`	`true` or `false`	When using the MAME core for the YM2612, this instructs the chip emulator to update the left/right channels alternatively, creating a nice pseudo-stereo effect. Default: `false`.

5.2.4 UI Config Block

Key Name	Type	Description
`animations-enabled`	`true` or `false`	When true, then various animations will be displayed by the user interface. This does not affect the VU meters. Default: ‘`true`’.

`banner`	One of: `graffiti`, `cyber`, `rounded`, `chunky`, `soft`, `doomed`, `stronger`, `speed`; or an array containing one or more of these values.	The banner to show at the top of the interface. If this is an array, a random banner from the array will be chosen. This cannot be an empty array. Default: an array with all of the possible values.
`banner-anim`	One of: `slide-in`, `dissolve`, `dissolve-from-nothing`, `wipe-from-left`, `wipe-from-right`	The type of banner animation to play when first starting Benben. This only has an effect when animations are enabled.
`theme`	A String or an Array of Strings	The theme to use for the UI. If this value is a string, then only that theme is used. If the value is an array, then a random theme will be chosen from the array values. Use the `--theme list` command line option to see a list of valid themes. Default: '`default`' (the default theme). See Chapter 6 [Themes], page 32,
`display-loop-times-separate`	`true` or `false`	When '`true`', the length of the loop segment will be shown in parentheses along side the track playback time in the song queue. Default: '`false`'.

5.2.5 Modules Config Block

Key Name	Type	Description
`default-panning`	An integer between 0 and 255	The default amount of panning for modules with no panning information. Default: '`69`'.
`interpolation`	One of: `nearest`, `linear`, `spline`	What kind of interpolation to apply to the sound. Default: '`linear`'.
`fade-out-songs`	`true` or `false`	When true, songs are looped and slowly faded out at the end. Otherwise songs just simply "end" when they're at the end. Default: '`false`'.

`post-track-seconds`	An integer between 0 and 255.	When you are not looping a module file, and reverb is enabled at the start of the track, then this many seconds of extra silence will added to the end of the song to allow the reverb tails to finish playing. Default: '`1`'.

5.2.6 MIDI Config Block

Key Name	Type	Description
`soundfont`	String	The path to the SoundFont file to use when playing MIDI/MUS files. Without a SoundFont, MIDI/MUS files cannot be played. Default: empty string.
`reverb-enabled`	`true` or `false`	When `true`, then a reverb effect will be enabled by default. This setting is specific to MIDI/MUS files since they handle reverb differently. Default: '`true`'.
`reverb-type`	One of: `mverb` or `zita`	The type of reverb to apply when reverb is enabled. This setting is specific to MIDI/MUS files since they handle reverb differently. Default: '`mverb`'.
`reverb-amount`	An integer between 0 and 255	The default amount of reverb to apply. MIDI/MUS files may still change this per-channel during playback. This setting is specific to MIDI/MUS files since they handle reverb differently. Default: '`64`'.
`disable-remapping`	`true` or `false`	Whether or not to attempt to map unknown instruments to a known instrument. Default: '`false`'.
`inst-reverb-level`	An integer between 0 and 65535, or `null`.	When this is an integer, then all SoundFont instruments will have their defined reverb level overridden with this level. Default: '`null`'.

`chorus-enabled`	`true` or `false`	When `true`, then a chorus effect will be enabled by default. This setting is specific to MIDI/MUS files since they handle chorus differently. Default: '`true`'.
`chorus-amount`	An integer between 0 and 255	The default amount of chorus to apply. MIDI/MUS files may still change this per-channel during playback. This setting is specific to MIDI/MUS files since they handle chorus differently. Default: '`0`'.
`chorus-interpolation`	One of: `linear`, `cubic`, `hermite`, `hermite-alt`, `parabolic`, `2x`, `optimal-2x-4p-2o`, or `optimal-2x-4p-4o`	The type of interpolation to use within the chorus effect. This setting is specific to MIDI/MUS files since they handle chorus differently. Default: '`linear`'.
`inst-chorus-level`	An integer between 0 and 65535, or `null`.	When this is an integer, then all SoundFont instruments will have their defined chorus level overridden with this level. Default: '`null`'.
`voice-filter-type`	One of: `standard`, `cem`, `ssm`, `hornet`, `ms20`, `disabled`	What type of filter to use for voices. `Standard` is the standard filter as described by the SoundFont specifications, and changing this value may cause your MIDI/MUS files to sound unintentionally different. Default: '`Standard`'.
`channel-filter-type`	One of: `standard`, `cem`, `ssm`, `hornet`, `ms20`, `disabled`	What type of filter to use for channels that request a lowpass filter to be applied during playback. `standard` is the standard filter as described by the SoundFont specifications, and changing this value may cause your MIDI/MUS files to sound unintentionally different. Default: '`standard`'.

`fadeout`	`true` or `false`	When `true`, then each MIDI/MUS will be looped and slowly faded out at the end. When `false`, then the songs simply "end" at the end. Default: '`false`'. Note that if this is `false`, then there is a small chance you will hear fragments or clicks at the end of some songs. This is because the length of time that audio plays from the MIDI (which is not necessarily the same as the length of a MIDI) may not be evenly divisible by the rendering buffer length.
`post-track-seconds`	An integer between 0 and 255	How much extra time to add at the end of a track. This is useful to hear the reverbs/instruments slowly fade out at the end rather than abruptly stop, so it's recommended that you leave it at the default or higher. Default: '`4`'.
`mverb-reverb-preset`	String	The preset to use for the reverb when reverb is enabled set to `mverb` in the `midi` block. This setting is specific to MIDI/MUS files since they handle reverb differently. Default: '`gm-default`'.
`zita-reverb-preset`	String	The preset to use for the reverb when reverb is enabled set to `zita` in the `midi` block. This setting is specific to MIDI/MUS files since they handle reverb differently. Default: '`gm-default`'.

5.2.7 C64 Config Block

Key Name	**Type**	**Description**
`song-length-db`	String	The path to the songlength's database file, as found on High Voltage Sid Collection (`https://www.hvsc.c64.org/`). Default: empty string (no database). See Chapter 4 [Playing SID Files], page 12,

`default-song-length`	An integer between 1 and 4294967295	The default amount of time in seconds to play a SID file when it is not found in the `song-length-db` database, or when no database is loaded. Default: '`180`'. See Chapter 4 [Playing SID Files], page 12,
`kernal-rom`	String	The path to a Commodore 64 KERNAL ROM file. Default: empty string. Note: the spelling is purposely "kernal". See Chapter 4 [Playing SID Files], page 12,
`basic-rom`	String	The path to a Commodore 64 BASIC ROM file. Default: empty string. See Chapter 4 [Playing SID Files], page 12,
`chargen-rom`	String	The path to a Commodore 64 Character (CHARGEN) ROM file. Default: empty string. See Chapter 4 [Playing SID Files], page 12,
`default-c64-model`	One of: `pal`, `ntsc`, `old-ntsc`, `drean`	The default model of Commodore 64 to emulate when a SID file does not specify a model. `pal` for European PAL-B model, `ntsc` for American/Japanese NTSC-M models, `old-ntsc` for NTSC-M models with old video chip, and `drean` for Argentinian PAL-N model. Default: '`ntsc`'.
`force-c64-model`	`true` or `false`	When `true`, then the `default-c64-model` value will always be applied regardless of what a SID file requests. Default: '`false`'.
`default-sid-model`	One of: `mos6581`, `mos8580`	The default model of SID chip to emulate when a SID file does not specify a model. Default: '`mos6581`'.

`force-sid-model`	`true` or `false`	When `true`, then the `default-sid-model` value will always be applied regardless of what a SID file requests. Default: '`false`'.
`sampling-method`	One of: `interpolate`, `resample-interpolate`	The resampling mode, where `interpolate` is faster while `resample-interpolate` is slower but more accurate. Default: '`interpolate`'.
`fast-sampling`	`true` or `false`	Whether or not a faster but slightly less accurate resampling method is used when `sampling-method` is `resample-interpolate`. Default: '`false`'.
`multi-sid-as-mono`	`true` or `false`	Whether or not to play multi-SID files (e.g. 2SID, 3SID) with one channel (monaural). Default: '`false`'. See Chapter 4 [Playing SID Files], page 12,

5.2.8 Equalizer Config Block

Key Name	Type	Description
`post-gain`	Float	The amount of gain to apply to the signal *after* the EQ has processed it, in decibels. Default: '`0.0`' (no change to the signal).
`low-shelf`	Sub-block. See Section 5.2.8.1 [Low-shelf Config Block], page 28,	The settings for the low shelf filter.
`bands`	An array of sub-blocks. See Section 5.2.8.2 [EQ Band Config Block], page 28,	An array of peaking EQ bands for the equalizer. There can be as many or as few as you wish.
`high-shelf`	Sub-block. See Section 5.2.8.3 [High-shelf Config Block], page 28,	The settings for the high shelf filter.

5.2.8.1 Low-shelf Config Block

Key Name	Type	Description
`frequency`	Float between 20.0 and 22050.0	The cutoff frequency for the low shelf in hertz. Default: '`80.0`'.
`gain`	Float	How much the volume of the signal under the low shelf is adjusted, in decibels. Default: '`0.0`'.
`bandwidth`	Float	How wide the transition band of the filter is, in octaves. Default: '`0.707`'.

5.2.8.2 EQ Band Config Block

Key Name	Type	Description
`frequency`	Float between 20.0 and 22050.0	The center frequency for the band in hertz.
`gain`	Float	How much the volume of the signal around the center frequency is adjusted, in decibels.
`bandwidth`	Float	The width of the band, in octaves.

5.2.8.3 High-shelf Config Block

Key Name	Type	Description
`frequency`	Float between 20.0 and 22050.0	The cutoff frequency for the high shelf in hertz. Default: '`80.0`'.
`gain`	Float	How much the volume of the signal under the high shelf is adjusted, in decibels. Default: '`0.0`'.
`bandwidth`	Float	How wide the transition band of the filter is, in octaves. Default: '`0.707`'.

5.2.9 ListenBrainz Config Block

Key Name	Type	Description
`token`	String or '`null`'.	Your user token for ListenBrainz. It's highly recommended that if you set this value, that you restrict read/write permissions for your configuration file to everyone except yourself. Default: '`null`'.
`vgm-prefer-english`	'`true`' or '`false`'	When '`true`', prefer to submit English GD3 tag data from VGM files when possible, falling back to Japanese tag data if it's available. Default: '`true`'.

5.3 Song-specific Config Files

In addition to the main config file, you can specify alternative configurations for specific song files or directories. These are matched using globbing[2]. When Benben notices you are playing a song file that matches one of these song-specific configurations, it will use the settings in that file to temporarily override your main configuration settings (the command line can still override everything, however). This lets you easily create settings specific to entire groups of songs. For example, I normally have the equalizer enabled, but I have specific settings for one album that disables the equalizer.

Song-specific config files reside in:

On Linux/Unix
: `$XDG_CONFIG_HOME/benben/song-configs/` (this usually means `~/.config/benben/song-configs/`).

The filename for every song-specific configuration file must start with `song-config-` and end in `.yaml`. For example, `song-config-x68k-eq.yaml`.

Song-specific config files follow the same basic format as the main configuration file, with a few changes. The most important change is that song-specific config files have an additional key, `match`, that lets you specify one or more patterns that will be used to match files. For example, the file `song-config-x68k-eq.yaml` on my system has this line at the very top:

```
match:
  - /mnt/nanako/vgms-and-mods/VGMs/X68000/**/*.vg?
```

This specifies that any VGM file in any subdirectory under `/mnt/nanako/vgms-and-mods/VGMs/X68000/` gets this song-specific config applied to it during playback. Any songs that don't match this pattern (or any pattern in any other song-specific config files I may have) use the settings in the main config file.

Any keys within a song-specific configuration file will override the main configuration settings (command line arguments still override everything). Keys that are missing from

[2] `https://en.wikipedia.org/wiki/Glob_(programming)`

the song configuration file will be filled in with the values from the main configuration file, or the defaults.

Song configuration files are almost identical to the normal configuration file format, except they ignore the following keys:

- `audio-driver`
- `alsa-device`
- `buffer-size`
- `sample-rate`
- `render-jobs`
- `tcp-driver-host`
- `tcp-driver-port`
- `tcp-driver-format`
- `recurse-directories`
- `repeat-playlist`
- `resampler`
- `remote`
- `seek-time`
- `equalizer`
- `ui`
- `listenbrainz`

All other keys are valid. See Section 5.2.1 [Main Configuration Options], page 16,

You can use `benben --dump-song-config` to print a new config to standard output that you can then modify as you wish.

5.4 Example Song-Specific Config File

```
match:
  - /mnt/nanako/vgms-and-mods/VGMs/X68000/**/*.vg?

equalizer:
  post-gain: 0.0
  low-shelf:
    frequency: 80.0
    gain: 2.7
    bandwidth: 1.8
  bands:
    - frequency: 105.0
      gain: 2.98
      bandwidth: 0.9
    - frequency: 275.0
      gain: -6.0
      bandwidth: 0.2
    - frequency: 6000.0
```

```
    gain: 1.5
    bandwidth: 1.0
high-shelf:
  frequency: 9001.0
  gain: 1.4
  bandwidth: 0.9
```

6 Themes

Benben stores its themes in YAML format[1] files within a special `themes` folder in its configuration directory. The exact location of this folder depends on your platform:

On Linux/Unix
: `$XDG_CONFIG_HOME/benben/themes/` (this usually means `~/.config/benben/themes/`).

Each theme is stored in its own file, and the filename of each theme must use the format `theme-<theme name>.yaml`. So for example, a theme named "dark-custom" must be in the `themes` directory and have the filename `theme-dark-custom.yaml`.

If this directory does not exist, Benben will create it at startup. If no theme is specified, Benben uses its default built-in theme.

6.1 Specifying a Theme

The theme to use is specified in your main configuration file in the `ui-config` section (See Chapter 5 [Configuration], page 14):

```
ui-config:
  theme: default
```

You can also change the theme on the command line:

```
benben --theme smooth-neon coolfile.mp3
```

The name of the theme depends on its filename. As mentioned previously, the filename of each theme must use the format `theme-<theme name>.yaml`. So for example, a theme stored in a file named `theme-dark-custom.yaml` is named "dark-custom" and can be specified using `benben --theme dark-custom`.

To see a list of available themes, use the `--list-themes` argument. See Chapter 3 [Command Line Options], page 7,

6.2 Theme File Format

Note that color values can be expressed multiple ways. See Section 6.2.1 [Theme Color Values], page 35,

Key Name	Type	Description
`version`	The integer value `1`	The version of the Theme Format specification this file conforms to. In all cases, this should be set to '`1`'. The default is '`1`' if this key is not specified.
`bg-color`	Color value	The color of the background.

[1] `https://en.wikipedia.org/wiki/YAML`

`fg-color`	Color value	The color of all text that isn't covered by another key. In other words, the default foreground color.
`banner-color`	Color value	The color of the banner text.
`banner-lines`	An array of color values	The colors of the lines above and below the banners. The last color is the one used when the banner is not animating. Note that the banner will cycle through these twice when animating. There can be a maximum of 15 colors, and there must be at least one.
`banner-fade-down-bright`	Color value	The color of bright lines when the banner text is doing its fade-down animation.
`banner-fade-down-dim`	Color value	The color of dim lines when the banner text is doing its fade-down animation.
`header-color`	Color value	The color of the field headers.
`err-color`	Color value	The color of the "Error:" text when displaying an error.
`cur-song-color`	Color value	The color of the currently playing song in the playback queue.
`prev-song-color`	Color value	The color of the previous song in the playback queue.
`next-song-color-1`	Color value	The color of the song that is next in the playback queue.
`next-song-color-2`	Color value	The color of the song that is two spots away in the playback queue.
`next-song-color-3`	Color value	The color of all songs three spots away and further in the playback queue.

`song-queue-box-color`	Color value	The color of the border of the playback queue box.
`song-queue-header-color`	Color value	The color of the "Song Queue" header text of the playback queue box.
`progress-bar-char`	A single character, or a string containing a single character.	The character used to draw the progress part (left-hand side) of the progress bar. This can be any UTF-8 encoded character as long as it's equivalent to a single Unicode code point (a single "character", essentially).
`progress-bar-space-char`	A single character, or a string containing a single character.	The character used to draw the right-hand side of the progress bar. This can be any UTF-8 encoded character as long as it's equivalent to a single Unicode code point (a single "character", essentially).
`progress-bar-colors`	An array of one or more color values	The colors for the progress bar. There must be at least one color, and there can be up to 38 different colors.
`vu-clip-color`	Color value	The color of the words "Left" and "Right" displayed next to the VU meters when a song clips.
`vu-clipped-channel-time`	An integer between 0 and 255, inclusive	How long in seconds the words "Left" and "Right" remain in their `vu-clip-color` when Benben detects that clipping has occurred.
`vu-bar-character`	A single character, or a string containing a single character.	The character used to draw the bar segments of the VU meter. This can be any UTF-8 encoded character as long as it's equivalent to a single Unicode code point (a single "character", essentially).

`vu-bar-character`	A single character, or a string containing a single character.	The character used to draw the tip of the VU meter. This can be any UTF-8 encoded character as long as it's equivalent to a single Unicode code point (a single "character", essentially).
`vu-colors`	An array of one or more color values	The colors for each VU meter segment. There must be at least one color, and there can be up to 70 different colors.

6.2.1 Color Values

Colors can be expressed in one of three ways:

- An integer between 0 and 255, inclusive. This acts as an index into the 8-bit ANSI color table[2], so for example a value of '`33`' would be a light blue color.
- An array of three integer values, each between 0 and 255, inclusive. This defines a 24-bit RGB color, where the first element is the red value, the second is the green, and the third is the blue.
- A string in the format `#RRGGBB`, which also defines a 24-bit RGB color. `RR` segment is a hex value between '`00`' and '`FF`', inclusive, that defines the red value, `GG` is a hex value for green, and `BB` is a hex value for blue. Note that the leading `#` is **required**, and so you'll need to enclose the entire value in double quotes.

Note that not all terminals support 24-bit RGB colors (or even ANSI colors). See Section 6.2.2 [Theme Troubleshooting], page 35,

6.2.2 My Theme Isn't Working!

Are you using 24-bit color values? If you are, and you're sure your terminal supports 24-bit ANSI colors, then it may be that the underlying S-Lang library just isn't detecting support for it properly. Try doing this before launching Benben:

```
export COLORTERM=truecolor
```

This will force it to treat the terminal as support 24-bit colors.

6.3 Example Theme File

```
bg-color: 0
fg-color: 250
header-color: 255
err-color: 196
cur-song-color: 201
prev-song-color: 242
next-song-color-1: 244
next-song-color-2: 239
next-song-color-3: 236
```

[2] `https://en.wikipedia.org/wiki/ANSI_escape_code#8-bit`

```
song-queue-box-color: 250
song-queue-header-color: 99

progress-bar-char: "*"
progress-bar-space-char: "-"
progress-bar-colors: [[170, 170, 170]]

vu-clip-color: 196
vu-clipped-channel-time: 1
vu-bar-character: "\u25A0"
vu-tip-character: "\u25B6"
vu-colors:
  - 105
  - 117
  - 119
  - 190
  - 206
  - 161
```

7 Remote Control

Benben can be controlled remotely using the "Benben Remote Protocol", which is fully implemented by a command line program called `remote-benben`. This program sends commands and receives responses from a running Benben instance, and allows a user to do things such as:

- Control Benben using the multimedia keys on your keyboard.
- Control Benben by connecting to the machine it's running on over SSH.
- Receive information from a running Benben instance and use that to automate other programs or display information elsewhere.
- Control Benben using Bluetooth devices such as headphones.

This is definitely not an exhaustive list of possible uses. Additionally, the protocol is fully open and can thus be implemented by other programs.

Communication between Benben and `remote-benben` (or other programs) happens over a UNIX domain socket. This is created by launching Benben with the `--remote` command line option, or by setting it up in the configuration file. The default location of the socket is `$XDG_DATA_HOME/benben/remote.sock`, and this also can be adjusted on the command line or via the configuration file. See Chapter 3 [Command Line Options], page 7,

Full technical documentation for the Benben Remote Protocol is available within Benben's repository: `https://chiselapp.com/user/MistressRemilia/repository/benben/file?name=remote-protocol.md&ci=tip`

7.1 Using The remote-benben Program

The `remote-benben` program allows full control of Benben. The program will connect to the default socket automatically. There is currently one option, `--socket`, which lets you connect to an alternate UNIX domain socket. You can get help by passing the `--help` option to the program.

The basic usage of `remote-benben` is:

```
remote-benben [options] <command>
```

To see a list of all commands, use this:

```
remote-benben cmd-help
```

Here are various examples of how to use the program. This is not an exhaustive list of commands:

```
$ remote-benben next  # Go to the next song

$ remote-benben prev  # Go to the previous song

$ remote-benben pause  # Toggle whether or not Benben is paused

$ remote-benben loop-up  # Increase number of times the song loops

$ remote-benben title  # Get the title of the currently playing track
```

7.2 remote-benben Commands

General Commands

`cmd-help` Displays a listing of all commands.

`help` Same as `cmd-help`.

`exit` Tells Benben to exit.

`ver` Returns Benben's version information.

`proto-ver`
Returns the version of the Benben Remote Protocol that's in use.

Control Commands

`next` Go to the next track.

`prev` Go to the previous track.

`ff` Seek forwards.

`rw` Seek backwards.

`lang` Toggles the displayed tag language (if applicable to the currently playing song).

`vol-up` Increases the volume.

`vol-down` Decreases the volume.

`pause` Toggles whether or not Benben is paused or playing.

`stop-after`
Toggles the "stop after current track" setting.

`pause-after`
Toggles the "pause after current track" setting.

Effect Commands

`eq` Toggles the equalizer on or off.

`soft-clip`
Toggles soft-clipping on or off.

`stereo` Toggles the stereo enhancer on or off.

`reverb` Toggles the reverb effect on or off.

`interp` Toggles the interpolation type (if the format supports interpolation).

`chorus` Toggles the chorus effect on or off (if the format supports it).

`dc-filter`
Toggles the DC offset filter on or off.

Looping Commands

`loop-up` Increase the number of times the song will loop.

`loop-down`
Decrease the number of times the song will loop.

`cur-loop` Returns the current playback loop.

`max-loops`
Returns the current maximum number of playback loops.

`repeat` Toggles repeating of the entire song queue.

Track Info Commands

`track` Returns the current track number (that is, the current track's position in the song queue).

`num-tracks`
Returns the total number of tracks in the song queue.

`format` Returns the current track's format.

`format-num`
Returns the current tracks' numerical format identifier.

`track-len`
Returns the length of the track.

`track-pos`
Returns the current playback position of the track.

`all-tracks`
Returns detailed information about all tracks in the song queue. The data returned is serialized using JSON.

Tag Info Commands

Not all formats have tag information, or have different information that what you can request. So, asking for the "genre" may actually return something other than the track's genre, depending on the format. For example, module files map the number of patterns to the genre field. The program understands this and will display the result correctly.

`title` Returns the song title.

`artist` Returns the track's artist.

`album` Returns the track's album.

`date` Returns the track's release date.

`genre` Returns the track's genre.

Other Commands

`stop-after?`
: Returns whether or not the "stop after current track" setting is currently enabled. A return value of `false` means that this is disabled.

`pause-after?`
: Returns whether or not the "pause after current track" setting is currently enabled. A return value of `false` means that this is disabled.

`status`
: Returns the status of the player. This may return one of the following values: `Paused` (Benben is paused), `Frame` (Benben is playing), `Fadeout` (Benben is playing and is fading out the current song), `Tails` (Benben is playing and is in-between tracks), `Done` (Benben is about to move to a new track).

8 Sending Audio Over TCP

Normally Benben plays audio by sending it to a backend "driver", such as PulseAudio or PortAudio, that communicates with your sound hardware. However, Benben is also able to send audio over a TCP socket, allowing audio to be piped over a network to another machine. Some backends such as PulseAudio can already do this, but Benben provides a more direct way of accomplishing this, with the trade-off of being somewhat less flexible in terms of realtime control.

To send audio over a TCP socket, you will need to use the TCP "driver". This can be selected on the command line using `--driver tcp`, or in the configuration file. The default behavior when using the TCP driver is to open a socket on `localhost` port `6969` and then listen for connections. Once a client connects, Benben begins playing and sending audio over the socket until it exits, or until the socket is closed.

The audio that is sent over TCP is the "raw" uncompressed audio that would normally be sent to your sound card; the default is to send it as 32-bit IEEE floating point samples. You may want to send a different format if you are limited on bandwidth, such as 16-bit signed PCM audio.

Full control of the TCP driver is accomplished either with the command line (See Chapter 3 [Command Line Options], page 7), or via the configuration file (See Chapter 5 [Configuration], page 14). It is especially powerful when combined with the `remote-benben` program. See Chapter 7 [Remote Control], page 37.

8.1 Sample Foramts

`float64` 64-bit IEEE floating point samples. Overkill in nearly all cases. This transmits eight bytes per sample of audio.

`float32` 32-bit IEEE floating point samples. The default format, and what Benben also uses when using other drivers. This transmits four bytes per sample of audio.

`int64` 64-bit signed integer PCM. Overkill in nearly all cases. This transmits eight bytes per sample of audio.

`int32` 32-bit signed integer PCM. This transmits four bytes per sample of audio.

`int24` 24-bit signed integer PCM. This transmits four bytes per sample of audio (not three).

`int16` 16-bit signed integer PCM. This is what CD audio uses. This transmits two bytes per sample of audio.

`int12` 12-bit signed integer PCM. This transmits two bytes per sample of audio.

`int8` 8-bit signed integer PCM. This transmits one byte per sample of audio.

`uint8` 8-bit unsigned integer PCM. This transmits one byte per sample of audio.

8.2 Example Using aplay As a Client Over SSH

One possible use of the TCP driver is to transmit audio from one machine to another, then play the audio using the `aplay` command on the other machine. This is a command that comes with ALSA, so it may not be available on non-Linux platforms, so you may need to adjust this example if you are on something like BSD.

This example assumes you have basic knowledge of SSH and how SSH port forwarding works, and are comfortable with the command line.

Let's assume we have two machines: computer1 and computer2. Benben is installed on computer1 and is configured to send audio over TCP to `localhost:6969`. We want to hear that audio on computer2. The first thing we want to do is connect to computer1 from computer2 using SSH and forwarding the proper port. So let's start on computer2:

```
[user@computer2]$ ssh -L 6969:computer1:6969 computer1
```

Next, on computer1, we launch Benben with the TCP driver. This will cause it to open a socket on computer1 port 6969:

```
[user@computer1]$ benben --driver tcp cool-song.mp3
```

Audio is now ready to be transmitted over port 6969. Since we've forwarded it over SSH, this port is now available on computer2. So, back on computer2, we then use netcat and `aplay` to receive the audio and play it. Note that we have to tell `aplay` the format, the number of channels, and the sample rate because this is raw audio that we're sending:

```
[user@computer2]$ nc localhost 6969 | aplay -f FLOAT_LE -c 2 -r 44100 -
```

At this point, the song will begin playing on computer1 and you'll hear it on computer2. You can control Benben directly on computer1, or use `remote-benben` via your SSH connection to control it. Or, you could even forward the UNIX domain socket over SSH and tell `remote-benben` to use the forwarded socket. At this point, the sky is the limit.

Index