harry

Claude Heiland-Allen

2019

Contents

1 COPYING.md

```
conveying.

An interactive user interface displays ”Appropriate Legal Notices” to
the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License. If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

#### 1. Source Code.

The ”source code” for a work means the preferred form of the work for
making modifications to it. ”Object code” means any non-source form of
a work.

A ”Standard Interface” means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

The ”System Libraries” of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form. A
”Major Component”, in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

The ”Corresponding Source” for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities. However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work. For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

The Corresponding Source need not include anything that users can
regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same
work.

#### 2. Basic Permissions.

All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met. This License explicitly affirms your unlimited
```

```
permission to run the unmodified Program. The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work. This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey,
without conditions so long as your license otherwise remains in force.
You may convey covered works to others for the sole purpose of having
them make modifications exclusively for you, or provide you with
facilities for running those works, provided that you comply with the
terms of this License in conveying all material for which you do not
control copyright. Those thus making or running the covered works for
you must do so exclusively on your behalf, under your direction and
control, on terms that prohibit them from making any copies of your
copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the
conditions stated below. Sublicensing is not allowed; section 10 makes
it unnecessary.

#### 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such
circumvention is effected by exercising rights under this License with
respect to the covered work, and you disclaim any intention to limit
operation or modification of the work as a means of enforcing, against
the work's users, your or third parties' legal rights to forbid
circumvention of technological measures.

#### 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

#### 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these
conditions:

-   a) The work must carry prominent notices stating that you modified
```

```
    it , and giving a relevant date .
-   b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under
    section 7. This requirement modifies the requirement in section 4
    to "keep intact all notices".
-   c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy. This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged. This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it .
-   d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit. Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

#### 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of
sections 4 and 5, provided that you also convey the machine-readable
Corresponding Source under the terms of this License, in one of these
ways:

-   a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.
-   b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the Corresponding
    Source from a network server at no charge.
-   c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source. This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.
-   d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge. You need not require recipients to copy the
    Corresponding Source along with the object code. If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source. Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.
-   e) Convey the object code using peer-to-peer transmission,
    provided you inform other peers where the object code and
    Corresponding Source of the work are being offered to the general
    public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal,
family, or household purposes, or (2) anything designed or sold for
incorporation into a dwelling. In determining whether a product is a
consumer product, doubtful cases shall be resolved in favor of
coverage. For a particular product received by a particular user,
"normally used" refers to a typical or common use of that class of
product, regardless of the status of the particular user or of the way
in which the particular user actually uses, or expects or is expected
to use, the product. A product is a consumer product regardless of
whether the product has substantial commercial, industrial or
non-consumer uses, unless such uses represent the only significant
mode of use of the product.

"Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to
install and execute modified versions of a covered work in that User
Product from a modified version of its Corresponding Source. The
information must suffice to ensure that the continued functioning of
the modified object code is in no case prevented or interfered with
solely because modification has been made.

If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information. But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or
updates for a work that has been modified or installed by the
recipient, or for the User Product in which it has been modified or
installed. Access to a network may be denied when the modification
itself materially and adversely affects the operation of the network
```

```
or violates the rules and protocols for communication across the
network.

Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

#### 7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law. If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it. (Additional permissions may be written to require their own
removal in certain cases when you modify the work.) You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders
of that material) supplement the terms of this License with terms:

-   a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or
-   b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or
-   c) Prohibiting misrepresentation of the origin of that material,
    or requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or
-   d) Limiting the use for publicity purposes of names of licensors
    or authors of the material; or
-   e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or
-   f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions
    of it) with contractual assumptions of liability to the recipient,
    for any liability that these contractual assumptions directly
    impose on those licensors and authors.

All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10. If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term. If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
```

```
not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions; the
above requirements apply either way.

#### 8. Termination.

You may not propagate or modify a covered work except as expressly
provided under this License. Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

However, if you cease all violation of this License, then your license
from a particular copyright holder is reinstated (a) provisionally,
unless and until the copyright holder explicitly and finally
terminates your license, and (b) permanently, if the copyright holder
fails to notify you of the violation by some reasonable means prior to
60 days after the cessation.

Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License. If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

#### 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run
a copy of the Program. Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance. However,
nothing other than this License grants you permission to propagate or
modify any covered work. These actions infringe copyright if you do
not accept this License. Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

#### 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License. You are not responsible
for enforcing compliance by third parties with this License.

```

```
An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations. If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License. For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

#### 11. Patents.

A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based. The
work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned
or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version. For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement). To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients. "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
```

```
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

A patent license is "discriminatory" if it does not include within the
scope of its coverage, prohibits the exercise of, or is conditioned on
the non-exercise of one or more of the rights that are specifically
granted under this License. You may not convey a covered work if you
are a party to an arrangement with a third party that is in the
business of distributing software, under which you make payment to the
third party based on the extent of your activity of conveying the
work, and under which the third party grants, to any of the parties
who would receive the covered work from you, a discriminatory patent
license (a) in connection with copies of the covered work conveyed by
you (or copies made from those copies), or (b) primarily for and in
connection with specific products or compilations that contain the
covered work, unless you entered into that arrangement, or that patent
license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

#### 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot convey a
covered work so as to satisfy simultaneously your obligations under
this License and any other pertinent obligations, then as a
consequence you may not convey it at all. For example, if you agree to
terms that obligate you to collect a royalty for further conveying
from those to whom you convey the Program, the only way you could
satisfy both those terms and this License would be to refrain entirely
from conveying the Program.

#### 13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the
Program, your modified version must prominently offer all users
interacting with it remotely through a computer network (if your
version supports such interaction) an opportunity to receive the
Corresponding Source of your version by providing access to the
Corresponding Source from a network server at no charge, through some
standard or customary means of facilitating copying of software. This
Corresponding Source shall include the Corresponding Source for any
work covered by version 3 of the GNU General Public License that is
incorporated pursuant to the following paragraph.

Notwithstanding any other provision of this License, you have
```

```
#### 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these
terms.

To do so, attach the following notices to the program. It is safest to
attach them to the start of each source file to most effectively state
the exclusion of warranty; and each file should have at least the
"copyright" line and a pointer to where the full notice is found.

        <one line to give the program's name and a brief idea of what it does.>
        Copyright (C) <year>  <name of author>

        This program is free software: you can redistribute it and/or modify
        it under the terms of the GNU Affero General Public License as
        published by the Free Software Foundation, either version 3 of the
        License, or (at your option) any later version.

        This program is distributed in the hope that it will be useful,
        but WITHOUT ANY WARRANTY; without even the implied warranty of
        MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
        GNU Affero General Public License for more details.

        You should have received a copy of the GNU Affero General Public License
        along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper
mail.

If your software can interact with users remotely through a computer
network, you should also make sure that it provides a way for users to
get its source. For example, if your program is a web application, its
interface could display a "Source" link that leads users to an archive
of the code. There are many ways you could offer source, and different
solutions will be better for different programs; see section 13 for
the specific requirements.

You should also get your employer (if you work as a programmer) or
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. For more information on this, and how to apply and follow
the GNU AGPL, see <https://www.gnu.org/licenses/>.
```

2 .gitignore

harry

3 harry.c

```
/*
harry -- text-mode audio file viewer
Copyright (C) 2019  Claude Heiland-Allen <claude@mathr.co.uk>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#define _GNU_SOURCE
#define _POSIX_C_SOURCE 199309

#include <math.h>
#include <stddef.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>

#include <getopt.h>
#include <ncurses.h>
#include <sndfile.h>

#ifdef WITH_SDL2
#include <SDL2/SDL.h>
#include <SDL2/SDL_audio.h>
#endif

#define FACTOR 1.0594630943592953

extern const char _binary_harry_c_start[], _binary_harry_c_end[];
extern const char _binary_Makefile_start[], _binary_Makefile_end[];
extern const char _binary_README_md_start[], _binary_README_md_end[];
extern const char _binary_COPYING_md_start[], _binary_COPYING_md_end[];

int write_file(const char *name, const char *start, const char *end)
{
  printf("writing '%s'... ", name);
  fflush(stdout);
  FILE *file = fopen(name, "wxb");
  if (! file)
  {
    printf("FAILED\n");
    return 0;
  }
  int ok = (fwrite(start, end - start, 1, file) == 1);
  fclose(file);
  if (ok)
    printf("ok\n");
  else
    printf("FAILED\n");
  return ok;
}

struct waveform
{
  struct waveform *next;
  struct waveform *prev;
  double samplerate;
  int channels;
  ssize_t frames;
  float *rms;
  float *avg;
  float *min;
  float *max;
  int borrowed;
};

struct waveform_list
{
  struct waveform head;
  struct waveform tail;
};

void free_waveform(struct waveform *w)
{
  if (! w) return;
  if (w->next) w->next->prev = w->prev;
  if (w->prev) w->prev->next = w->next;
  w->next = 0;
  w->prev = 0;
  if (w->rms) free(w->rms);
  if (! w->borrowed)
  {
    if (w->avg) free(w->avg);
    if (w->min) free(w->min);
    if (w->max) free(w->max);
  }
  w->rms = 0;
  w->avg = 0;
  w->min = 0;
  w->max = 0;
  free(w);
}

void free_waveform_list(struct waveform_list *l)
{
  if (! l) return;
  while (l->head.next->next) free_waveform(l->head.next);
  free(l);
```

```c
}

struct waveform *read_sound_file(const char *name)
{
  struct waveform *w = calloc(1, sizeof(*w));
  if (! w)
  {
    return 0;
  }
  SF_INFO info;
  memset(&info, 0, sizeof(info));
  SNDFILE *ifile = sf_open(name, SFM_READ, &info);
  if (! ifile)
  {
    free(w);
    return 0;
  }
  w->samplerate = info.samplerate;
  w->channels = info.channels;
  w->frames = info.frames;
  ssize_t bytes = sizeof(float) * (w->frames + (w->frames & 1)) * w->channels;
  w->rms = malloc(bytes);
  w->avg = w->rms;
  w->min = w->rms;
  w->max = w->rms;
  w->borrowed = 1;
  if (w->frames & 1)
    for (int channel = 0; channel < w->channels; ++channel)
      w->rms[w->frames * w->channels + channel] = 0;
  if (w->rms && w->frames == sf_readf_float(ifile, w->rms, w->frames))
  {
    sf_close(ifile);
    return w;
  }
  else
  {
    sf_close(ifile);
    free_waveform(w);
    return 0;
  }
}

struct waveform_list *singleton(struct waveform *base)
{
  if (! base)
  {
    return 0;
  }
  struct waveform_list *l = calloc(1, sizeof(*l));
  if (! l)
  {
    free_waveform(base);
    return 0;
  }
  l->head.next = base;
  l->tail.prev = base;
  base->next = &l->tail;
  base->prev = &l->head;
  return l;
}

struct waveform_list *build_mipmaps(struct waveform_list *l)
{
  if (! l) return 0;
  while (l->head.next->frames > 1)
  {
    struct waveform *u = l->head.next;
    struct waveform *v = calloc(1, sizeof(*v));
    if (! v)
    {
      free_waveform_list(l);
      return 0;
    }
    v->samplerate = u->samplerate / 2;
    v->channels = u->channels;
    v->frames = (u->frames + 1) / 2;
    ssize_t bytes = sizeof(float) * (v->frames + (v->frames & 1)) * v->channels;
    v->rms = malloc(bytes);
    v->avg = malloc(bytes);
    v->min = malloc(bytes);
    v->max = malloc(bytes);
    if (v->rms && v->min && v->max)
    {
      #pragma omp parallel for schedule(static)
      for (ssize_t i = 0; i < v->frames + (v->frames & 1); ++i)
      {
        for (ssize_t c = 0; c < v->channels; ++c)
        {
          float mi = 1.0 / 0.0;
          float ma = -1.0 / 0.0;
          float av = 0;
          float s2 = 0;
          float s0 = 0;
          for (ssize_t j = 2 * i; j < 2 * i + 2 && j < u->frames; ++j)
          {
            ssize_t ix = j * u->channels + c;
            mi = fminf(mi, u->min[ix]);
            ma = fmaxf(ma, u->max[ix]);
            av = av + u->avg[ix];
            s2 = s2 + u->rms[ix] * u->rms[ix];
            s0 = s0 + 1;
          }
          ssize_t ix = i * v->channels + c;
          v->min[ix] = mi;
          v->max[ix] = ma;
          v->avg[ix] = av / s0;
          v->rms[ix] = sqrtf(s2 / s0);
        }
      }
    }
    else
    {
      free_waveform(v);
      free_waveform_list(l);
```

```
        return 0;
      }
      v->next = l->head.next;
      v->prev = &l->head;
      l->head.next->prev = v;
      l->head.next = v;
    }
    return l;
}

struct cursor
{
  struct waveform *wave;
  double offset;
  double loggain;
  double logspeed;
};

int cursor_up(struct cursor *c, int width)
{
  (void) width;
  if (c->wave->next->next)
  {
    c->wave = c->wave->next;
    c->offset *= 2;
    if (c->offset > c->wave->frames)
      c->offset = c->wave->frames;
    return 1;
  }
  return 0;
}

void cursor_end(struct cursor *c, int width)
{
  while (cursor_up(c, width))
    ;
}

int cursor_down(struct cursor *c, int width)
{
  if (c->wave->prev->prev && c->wave->prev->frames >= width / 4)
  {
    c->wave = c->wave->prev;
    c->offset /= 2;
    if (c->offset < 0)
      c->offset = 0;
    return c->wave->frames >= width / 2;
  }
  return 0;
}

void cursor_home(struct cursor *c, int width)
{
  cursor_up(c, width);
  while (cursor_down(c, width))
    ;
}
```

```
void cursor_left(struct cursor *c, int width)
{
  c->offset -= width / 4.0;
  if (c->offset <= 0)
  {
    c->offset = 0;
  }
}

void cursor_right(struct cursor *c, int width)
{
  c->offset += width / 4.0;
  if (c->offset >= c->wave->frames)
  {
    c->offset = c->wave->frames;
  }
}

struct cursor *cursor_init(struct waveform_list *l, int width)
{
  if (width < 1) return 0;
  struct cursor *c = calloc(1, sizeof(*c));
  if (! c) return 0;
  c->wave = l->tail.prev;
  c->offset = 0;
  c->loggain = 0;
  c->logspeed = 0;
  cursor_home(c, width);
  return c;
}

void printw_time(double ms)
{
  int milliseconds = floor(fmod(ms, 1000.0));
  int seconds = floor(fmod(ms, 1000.0 * 60) / 1000.0);
  int minutes = floor(fmod(ms, 1000.0 * 60 * 60) / (1000 * 60));
  int hours = floor(fmod(ms, 1000.0 * 60 * 60 * 24) / (1000 * 60 * 60));
  int days = floor(ms / (1000.0 * 60 * 60 * 24));
  printw
    ( "%d:%02d:%02d:%02d.%03d"
    , days, hours, minutes, seconds, milliseconds
    );
}

struct audio
{
  struct waveform_list *waveforms;
  double offset;
  int loggain;
  double gain;
  double increment;
  int floor_log2_increment;
  double fract_log2_increment;
  double factor;
  int playing;
  int looping;
  int mute;
  int ab;
  double loop_start;
  double loop_end;
  int follow;
};

void display(struct cursor *c, const char *name, struct audio *a)
{
  static int seed = 147;
  srand(seed++);
  int row = 0, col = 0;
  getmaxyx(stdscr, row, col);
  int n = c->wave->channels;
  int y0 = (row - 1) % (n * 2);
  row -= y0 + 1;
  float gain = pow(FACTOR, c->loggain);
  int first = 1;
  double start = -1.0 / 0.0;
  double end = 1.0 / 0.0;
  for (int i = 0; i <= col; ++i)
  {
    double k = c->offset + i - col / 2;
    if (0 <= k && k < c->wave->frames)
    {
      if (first)
      {
        start = k;
        first = 0;
      }
      end = k;
      for (int channel = 0; channel < n; ++channel)
      {
        ssize_t ix = ((ssize_t)(floor(k))) * n + channel;
        float avg = c->wave->avg[ix] * gain;
        float rms = c->wave->rms[ix] * gain;
        float dev = sqrtf(fmaxf(0, rms * rms - avg * avg));
        float min = c->wave->min[ix] * gain;
        float max = c->wave->max[ix] * gain;
        if (min > 0 || max < 0) rms = 0;
        float ys[] = { min, max, avg - dev, avg + dev, avg, 0 };
        for (int l = 0; l < 6; ++l)
        {
          float y = -ys[l];
          y = fminf(y, 1);
          y = fmaxf(y, -1);
          // -1 .. 1
          y /= 2;
          y += 0.5;
          // 0 .. 1
          y += channel;
          // 0 .. channels
          y /= n;
          y *= row;
          // dither
          y += rand() / (double) RAND_MAX - 0.5;
          // 0 .. row
          y = roundf(y);
          if (y < 0) y = 0;
          if (y > row - 1) y = row - 1;
          ys[1] = y + y0;
        }
        min = fminf(ys[0], ys[1]);
        max = fmaxf(ys[0], ys[1]);
        float devi = fminf(ys[2], ys[3]);
        float deva = fmaxf(ys[2], ys[3]);
        avg = ys[4];
        mvaddch(min, i, '^' | A_DIM);
        for (int j = min + 1; j < max; ++j)
          mvaddch(j, i, '+' | A_DIM);
        mvaddch(max, i, 'v' | A_DIM);
        if (min <= devi && deva <= max)
        {
          mvaddch(devi, i, '~' | A_BOLD);
          for (int j = devi + 1; j < deva; ++j)
            mvaddch(j, i, '*' | A_BOLD);
          mvaddch(deva, i, '~' | A_BOLD);
        }
        mvaddch(ys[4], i, '@' | A_BOLD);
        if (ys[4] != ys[5])
          mvaddch(ys[5], i, '-');
      }
    }
    if (a->follow && i == col / 2)
      for (int j = 0; j < row; ++j)
        mvaddch(y0 + j, i, '|');
  }
  if (a->ab > 0)
  {
    double k = a->loop_start * c->wave->frames / a->waveforms->tail.prev->frames;
    double i = k + col / 2 - c->offset;
    if (0 <= i && i < col)
    {
      int ii = floor(i);
      for (int j = 0; j < row; ++j)
        mvaddch(y0 + j, ii, ':' | A_BOLD);
    }
  }
  if (a->ab > 1)
  {
    double k = a->loop_end * c->wave->frames / a->waveforms->tail.prev->frames;
    double i = k + col / 2 - c->offset;
    if (-1 < i && i <= col - 1)
    {
      int ii = ceil(i);
      for (int j = 0; j < row; ++j)
        mvaddch(y0 + j, ii, ':' | A_BOLD);
    }
  }
  if (! a->follow)
  {
    double k = a->offset * c->wave->frames / a->waveforms->tail.prev->frames;
    double i = k + col / 2 - c->offset;
    if (-0.5 <= i && i < col + 0.5)
    {
      int ii = round(i);
      for (int j = 0; j < row; ++j)
        mvaddch(y0 + j, ii, '|' | A_BOLD);
    }
  }
  mvprintw(0, 0, "file ");
  printw_time(c->wave->frames * 1000.0 / c->wave->samplerate);
  printw(" %s", name);
  mvprintw(1, 0, "view ");
  printw_time(col * 1000.0 / c->wave->samplerate);
  printw
    ( " gain %g | speed %g"
    , pow(FACTOR, c->loggain)
    , pow(FACTOR, c->logspeed)
    );
  if (a->ab == 1) printw(" | A");
  if (a->ab == 2) printw(" | A-B");
  if (a->looping) printw(" | loop");
  if (a->playing) printw(" | play");
  if (a->mute) printw(" | mute");
  mvprintw(2, 0, "at   ");
  printw_time(c->offset * 1000.0 / c->wave->samplerate);
  {
    int col0 = floor(col * fmin(fmax(0, start / c->wave->frames), 1));
    int col1 = ceil (col * fmin(fmax(0, end   / c->wave->frames), 1));
    for (int i = 0; i < col0; ++i)
      mvaddch(y0 + row, i, '[' | A_DIM);
    for (int i = col0; i < col1; ++i)
      mvaddch(y0 + row, i, '=');
    for (int i = col1; i < col; ++i)
      mvaddch(y0 + row, i, ']' | A_DIM);
  }
  {
    double len = a->waveforms->tail.prev->frames;
    int start = floor(col * a->loop_start / len);
    int end   = floor(col * a->loop_end   / len);
    if (a->ab > 0)
    {
      mvaddch(y0 + row, start, '|' | A_BOLD);
      mvaddch(y0 + row, start + 1, ':' | A_BOLD);
    }
    if (a->ab > 1)
    {
      mvaddch(y0 + row, end, ':' | A_BOLD);
      mvaddch(y0 + row, end + 1, '|' | A_BOLD);
    }
    mvaddch(y0 + row , floor(col * a->offset / len), '>' | A_BOLD);
  }
}

void exit_cb(void)
{
  endwin();
}
```

```
void audio_speed(struct audio *a, double speed)
{
  a->increment = speed / a->factor;
  double l = log2(a->increment);
  a->floor_log2_increment = floor(l);
  a->fract_log2_increment = l - floor(l);
}

#ifdef WITH_SDL2

static inline float tabread4
  ( const float *buffer, ssize_t l, ssize_t channels
  , ssize_t channel, double d
  )
{
  ssize_t d1 = floor(d);
  ssize_t d0 = d1 - 1;
  ssize_t d2 = d1 + 1;
  ssize_t d3 = d1 + 2;
  double t = d - d1;
  d0 = (0 <= d0 && d0 < l) ? d0 : 0;
  d1 = (0 <= d1 && d1 < l) ? d1 : d0;
  d2 = (0 <= d2 && d2 < l) ? d2 : d1;
  d3 = (0 <= d3 && d3 < l) ? d3 : d2;
  double y0 = buffer[channels * d0 + channel];
  double y1 = buffer[channels * d1 + channel];
  double y2 = buffer[channels * d2 + channel];
  double y3 = buffer[channels * d3 + channel];
  double a0 = -t*t*t + 2*t*t - t;
  double a1 = 3*t*t*t - 5*t*t + 2;
  double a2 = -3*t*t*t + 4*t*t + t;
  double a3 = t*t*t - t*t;
  return (a0 * y0 + a1 * y1 + a2 * y2 + a3 * y3) / 2;
}

static inline void audio1(struct audio *a, float *out, ssize_t channels)
{
  if (a->increment != 0)
  {
    int l = a->floor_log2_increment;
    float f = a->fract_log2_increment;
    int l1 = l + 1;
    float f1 = 1 - f;
    const struct waveform *base = a->waveforms->tail.prev;
    for (int i = 0; i < l && base->prev->prev; ++i)
      base = base->prev;
    double base_offset = a->offset / pow(2, fmax(0, l));
    const struct waveform *next = base->prev;
    double next_offset = a->offset / pow(2, l1);
    if (l + f > 0 && next->prev)
    {
      for (int channel = 0; channel < channels; ++channel)
        out[channel] = (! a->mute) * a->gain *
          ( f1 * tabread4
            ( base->avg, base->frames, base->channels
            , channel, base_offset
            )
          + f  * tabread4
            ( next->avg, next->frames, next->channels
            , channel, next_offset
            )
          );
    }
    else
    {
      for (int channel = 0; channel < channels; ++channel)
        out[channel] = (! a->mute) * a->gain * tabread4
          ( base->avg, base->frames, base->channels
          , channel, base_offset
          );
    }
    double inc = a->increment;
    a->offset += inc;
    if (a->ab == 2)
    {
      double len = a->waveforms->tail.prev->frames;
      double start = a->loop_start;
      double end = a->loop_end;
      if (end < start) end += len;
      if (a->offset < start) a->offset += len;
      a->offset = fmod(a->offset - start, end - start) + start;
      a->offset = fmod(a->offset, len);
    }
    else if (a->offset >= a->waveforms->tail.prev->frames)
    {
      if (a->looping)
      {
        a->offset = fmod(a->offset, a->waveforms->tail.prev->frames);
      }
      else
      {
        a->playing = 0;
        audio_speed(a, 0);
      }
    }
  }
  else
  {
    for (ssize_t c = 0; c < channels; ++c)
    {
      out[c] = 0;
    }
  }
}

void audio_cb(void *userdata, Uint8 *stream, int len)
{
  struct audio *a = userdata;
  ssize_t c = a->waveforms->tail.prev->channels;
  float *b = (float *) stream;
  ssize_t m = len / sizeof(float) / c;
  ssize_t k = 0;
  for (ssize_t i = 0; i < m; ++i)
  {
    float out[c];
    audio1(a, out, c);
    for (int j = 0; j < c; ++j)
    {
      b[k++] = out[j];
    }
  }
}

#endif

enum audio_device
  { audio_null
#ifdef WITH_SDL2
  , audio_sdl
#endif
  };

int main(int argc, char **argv)
{
  enum audio_device adev = audio_null;
  (void) adev;
#ifdef WITH_SDL2
  adev = audio_sdl;
#endif
  while (1)
  {
    int option_index = 0;
    static struct option long_options[] =
      { { "audio",   required_argument, 0, 'a' }
      , { "help",    no_argument,       0, 'h' }
      , { "version", no_argument,       0, 'v' }
      , { "source",  no_argument,       0, 'S' }
      , { 0,         0,                 0,  0  }
      };
    int opt = getopt_long(argc, argv, "a:hH?vVS", long_options, &option_index);
    if (opt == -1) break;
    switch (opt)
    {
      case 'a':
        if (0 == strcmp(optarg, "null")) adev = audio_null;
#ifdef WITH_SDL2
        else
        if (0 == strcmp(optarg, "sdl")) adev = audio_sdl;
#endif
        else
        {
          fprintf(stderr, "%s: error: unknown audio driver '%s'\n", argv[0], optarg);
          return 1;
        }
        break;
      case 'h':
      case 'H':
      case '?':
        printf(
          "harry -- text-mode audio file viewer\n"
          "Copyright (C) 2019  Claude Heiland-Allen\n"
          "License: GNU AGPLv3+\n"
          "\n"
          "Usage:\n"
          "    %s [-audio <driver>] FILE\n"
          "        open audio file for interactive viewing\n"
          "        this harry is compiled with these audio drivers:\n"
#ifdef WITH_SDL2
          "            null\n"
          "            sdl (default)\n"
#else
          "            null (default)\n"
#endif
          "    %s -?,-h,-H,--help\n"
          "        this message\n"
          "    %s -v,-V,--version\n"
          "        output version string\n"
          "    %s -S,--source\n"
          "        output %s's source code to the current working directory\n"
          "        files written: harry.c Makefile README.md COPYING.md\n"
          "\n"
          "Keys:\n"
          "    ESC, Q         quit\n"
          "    Left, Right    scroll through time\n"
          "    Up, Down       zoom in and out\n"
          "    Home           zoom out to fit whole file in view\n"
          "    End            zoom in to single samples\n"
          "    +, =, -, 1     adjust audio display volume\n"
          "    9, 0           adjust audio output volume\n"
          "    M              toggle mute\n"
          "    Space, P       toggle playback\n"
          "    [, ]           adjust playback speed\n"
          "    Shift-L        toggle looping\n"
          "    l              configure A-B looping\n"
          "    F              toggle playback cursor follow mode\n"

          , argv[0], argv[0], argv[0], argv[0], argv[0]
          );
        return 0;
      case 'v':
      case 'V':
        printf("%d\n", 2);
        return 0;
      case 'S':
      {
        int ok = 1;
        ok &= write_file("harry.c", _binary_harry_c_start, _binary_harry_c_end);
        ok &= write_file("Makefile", _binary_Makefile_start, ↙
            ↳ _binary_Makefile_end);
        ok &= write_file("README.md", _binary_README_md_start, ↙
            ↳ _binary_README_md_end);
        ok &= write_file("COPYING.md", _binary_COPYING_md_start, ↙
            ↳ _binary_COPYING_md_end);
        return ! ok;
      }
    }
  }
  if (optind >= argc)
  {
    fprintf(stderr, "%s: error: missing argument\n", argv[0]);
    return 1;
  }
  int row = 0, col = 0;
  initscr();
  atexit(exit_cb);
  cbreak();
  keypad(stdscr, TRUE);
  noecho();
  curs_set(0);
  const char *msg = "...loading...";
  getmaxyx(stdscr, row, col);
  mvprintw(row / 2, (col -  strlen(msg)) / 2, "%s", msg);
  refresh();
  struct waveform_list *l =
    build_mipmaps(singleton(read_sound_file(argv[optind])));
  if (! l)
  {
    endwin();
    fprintf(stderr, "%s: error: failed to load '%s'\n", argv[0], argv[optind]);
    return 1;
  }
  struct audio audio = { 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1 };
  int audio_running = 0;
#ifdef WITH_SDL2
  SDL_AudioDeviceID dev = 0;
  if (adev == audio_sdl)
  {
    SDL_Init(SDL_INIT_AUDIO);
    SDL_AudioSpec want, have;
    want.freq = l->tail.prev->samplerate;
    want.format = AUDIO_F32;
    want.channels = l->tail.prev->channels;
    want.samples = 4096;
    want.callback = audio_cb;
    want.userdata = &audio;
    dev = SDL_OpenAudioDevice(NULL, 0, &want, &have, SDL_AUDIO_ALLOW_ANY_CHANGE)↙
        ↳ ;
    if (dev)
    {
      if (have.format != AUDIO_F32 || have.channels != want.channels)
      {
        endwin();
        fprintf(stderr, "%s: error: bad audio parameters\n", argv[0]);
        fprintf
          ( stderr, "want: %d %d %d %d\n"
          , want.freq, want.format, want.channels, want.samples
          );
        fprintf
          ( stderr, "have: %d %d %d %d\n"
          , have.freq, have.format, have.channels, have.samples
          );
      }
      else
      {
        audio.factor = have.freq / l->tail.prev->samplerate;
        SDL_PauseAudioDevice(dev, 0);
        audio_running = 1;
      }
    }
  }
#endif
  struct cursor *c = cursor_init(l, col);
  int running = TRUE;
  struct timespec last, now;
  while (running)
  {
    erase();
    display(c, argv[optind], &audio);
    refresh();
    getmaxyx(stdscr, row, col);
    switch (getch())
    {
      case KEY_UP: cursor_up(c, col); break;
      case KEY_DOWN: cursor_down(c, col); break;
      case KEY_LEFT:
        cursor_left(c, col);
        if (audio.follow)
          audio.offset = c->offset / c->wave->samplerate * l->tail.prev->↙
              ↳ samplerate;
        break;
      case KEY_RIGHT:
        cursor_right(c, col);
        if (audio.follow)
          audio.offset = c->offset / c->wave->samplerate * l->tail.prev->↙
              ↳ samplerate;
        break;
      case KEY_HOME:
        cursor_home(c, col);
        if (audio.follow)
          audio.offset = c->offset / c->wave->samplerate * l->tail.prev->↙
              ↳ samplerate;
        break;
      case KEY_END:
        cursor_end(c, col);
        if (audio.follow)
          audio.offset = c->offset / c->wave->samplerate * l->tail.prev->↙
              ↳ samplerate;
        break;
      case ' ':
      case 'P':
      case 'p':
        audio.playing = ! audio.playing;
        if (audio.playing) {
          halfdelay(1);
          clock_gettime(CLOCK_MONOTONIC, &last);
        }
        audio_speed(&audio, audio.playing * pow(FACTOR, c->logspeed));
        break;
      case 'F':
      case 'f': audio.follow = ! audio.follow; break;
      case 'L': audio.looping = ! audio.looping; break;
      case 'l':
        switch (audio.ab)
        {
          case 0: audio.loop_start = audio.offset; audio.ab = 1; break;
          case 1: audio.loop_end = audio.offset; audio.ab = 2; break;
          case 2: audio.ab = 0; break;
        }
        break;
      case '[':
        c->logspeed -= 1;
        audio_speed(&audio, audio.playing * pow(FACTOR, c->logspeed));
        break;
      case ']':
        c->logspeed += 1;
        audio_speed(&audio, audio.playing * pow(FACTOR, c->logspeed));
        break;
      case 10:
        c->logspeed = 0;
        audio_speed(&audio, audio.playing * pow(FACTOR, c->logspeed));
        break;
      case '9':
        audio.loggain -= 1;
        audio.gain = pow(FACTOR, audio.loggain);
        break;
      case '0':
        audio.loggain += 1;
        audio.gain = pow(FACTOR, audio.loggain);
        break;
      case 'M': case 'm': audio.mute = ! audio.mute; break;
      case '+': case '=': c->loggain += 1; break;
      case '-': c->loggain -= 1; break;
      case '1': c->loggain = 0; break;
      case 27: case 'Q': case 'q': running = FALSE; break;
    }
    if (audio.playing)
    {
      if (audio_running)
      {
        if (audio.follow)
        {
          double offset = audio.offset;
          c->offset = offset / l->tail.prev->samplerate * c->wave->samplerate;
        }
      }
      else
      {
        clock_gettime(CLOCK_MONOTONIC, &now);
        double dt = (now.tv_sec - last.tv_sec)
                  + (now.tv_nsec - last.tv_nsec) / 1.0e9;
        last.tv_sec = now.tv_sec;
        last.tv_nsec = now.tv_nsec;
        c->offset += pow(FACTOR, c->logspeed) * c->wave->samplerate * dt;
        if (c->offset >= c->wave->frames)
        {
          if (audio.looping)
          {
            c->offset = fmod(c->offset, c->wave->frames);
          }
          else
          {
            c->offset = c->wave->frames;
            audio.playing = FALSE;
            nocbreak();
            cbreak();
            audio_speed(&audio, audio.playing * pow(FACTOR, c->logspeed));
          }
        }
      }
    }
    else
    {
      nocbreak();
      cbreak();
    }
  }
#ifdef WITH_SDL2
  if (adev == audio_sdl)
  {
    SDL_CloseAudioDevice(dev);
    SDL_Quit();
  }
#endif
  free_waveform_list(l);
  free(c);
  return 0;
}
```

4 Makefile

```
#!/usr/bin/env -S make -f
# harry -- text-mode audio file viewer
# Copyright (C) 2019  Claude Heiland-Allen <claude@mathr.co.uk>
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU Affero General Public License as
# published by the Free Software Foundation, either version 3 of the
# License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Affero General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.

C99FLAGS = -std=c99 -Wall -Wextra -pedantic
OFLAGS = -O3
LIBS = -lsndfile -lncurses -lm
SOURCE = -Wl,harry.c -Wl,Makefile -Wl,README.md -Wl,COPYING.md
EMBED = -Wl,--format=binary $(SOURCE) -Wl,--format=default
OMP ?= -fopenmp
SDL ?= -DWITH_SDL2 `sdl2-config --cflags --libs`

```

```
harry: harry.c Makefile README.md COPYING.md
        gcc $(C99FLAGS) $(OFLAGS) $(OMP) -o harry harry.c $(LIBS) $(EMBED) $(SDL↙
           ↪ )
```

5 README.md

```
---
title: harry -- text-mode audio file viewer
author: Claude Heiland-Allen
date: 2019-11-14
---

# harry

text-mode audio file viewer

<https://mathr.co.uk/harry>

## source code

<https://code.mathr.co.uk/harry>

    git clone https://code.mathr.co.uk/harry.git

## dependencies

required

- gcc
- make
- libncurses-dev
- libsndfile-dev

optional

- libsdl2-dev

## build

with SDL audio backend and OpenMP parallism:

    make

without SDL:

    make SDL=

without OpenMP parallelism:

    make OMP=

## help

    harry -- text-mode audio file viewer
    Copyright (C) 2019  Claude Heiland-Allen
    License: GNU AGPLv3+
    Usage:
        harry [-audio <driver>] FILE
            open audio file for interactive viewing
            this harry is compiled with these audio drivers:
                null
                sdl (default)
        harry -?,-h,-H,--help
            this message
        harry -v,-V,--version
            output version string
        harry -S,--source
            output harry's source code to the current working directory
            files written: harry.c Makefile README.md COPYING.md

    Keys:
        ESC, Q          quit
        Left, Right     scroll through time
        Up, Down        zoom in and out
        Home            zoom out to fit whole file in view
        End             zoom in to single samples
        +, =, -, 1      adjust audio display volume
        9, 0            adjust audio output volume
        M               toggle mute
        Space, P        toggle playback
        [, ]            adjust playback speed
        Shift-L         toggle file looping
        l               configure A-B looping
        F               toggle playback cursor follow mode

## display

harry does mipmap reduction for more-correct display of resampled signals.
harry displays signal statistics per cell column, averaged over duration:

- ‘v‘: minimum
- ‘^‘: maximum
- ‘@‘: average
- ‘~‘: deviation
- ‘-‘: zero

A-B loop points are marked with ‘:‘, the play cursor is marked with ‘|‘.

At the bottom is a representation of the whole file: outside the view
is ‘[‘ and ‘]‘, inside the view is ‘=‘, loop points are marked ‘|:‘ and
‘:|‘.  The play cursor is marked ‘>‘.

## sound

harry plays sound using SDL2, which has a PulseAudio backend.  You can
run harry remotely and forward the audio over an SSH tunnel to play from
a local audio device:

    localhost$ scp ~/.config/pulse/cookie remotehost:.config/pulse/cookie
    localhost$ ssh -R 24713:localhost:4713 remotehost
    remotehost$ export PULSE_SERVER="tcp:localhost:24713"
    remotehost$ harry audio.wav
```

```
The remote host does not need PulseAudio daemon running.

## bugs

- harry eats a huge amount of RAM, about 20 bytes per sample per channel
  (compare with 2 bytes per sample per channel for 16bit WAV).
  This means typical audio files consume about 1GB for every 10 minutes.

## changelog

1. initial release

2. reduced memory consumption to a factor of 5 / 8

   fix use of uninitialized memory

   close SDL audio device when done

   free memory before exit

## legal

   harry -- text-mode audio file viewer
   Copyright (C) 2019  Claude Heiland-Allen <claude@mathr.co.uk>

   This program is free software: you can redistribute it and/or modify
   it under the terms of the GNU Affero General Public License as
   published by the Free Software Foundation, either version 3 of the
   License, or (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU Affero General Public License for more details.

   You should have received a copy of the GNU Affero General Public License
   along with this program.  If not, see <https://www.gnu.org/licenses/>.
```