Kitlist 2

Building

If building from the git repository:

```
$ aclocal
$ autoheader
$ autoreconf --install
$ automake --add-missing --copy
```

Thereafter, or if using the distributed tarball:

```
$ ./configure
$ make
$ sudo make install
```

When compiling on macOS with Clang, you may want to disable the following compiler warning:

```
./configure CXXFLAGS='-Wno-writable-strings'
```

When building from source from the git repository, you may get an error similar to the following during `make`:

```
kitlist.texi:7: @include: could not find version.texi
```

Run `make dist` which will re-create `version.texi`.

Build with wxWidgets debug enabled:

```
$ ./configure CPPFLAGS='-DwxDEBUG_LEVEL=1'
```

Build for development:

```
$ ./configure --enable-assertions --enable-maintainer-mode
```

During development you may also like to include the following flags to include more warnings:

```
CXXFLAGS=-Wall -Wextra
```

Distribution Check

```
$ make distcheck
```

Development Builds

On macOS install the following ports using MacPorts:

- wxWidgets
- pugixml

Note: wxWidgets should be compiled with `clang` on macOS. Compiling with GCC may cause runtime failures.

During development build with the following options:

See https://best.openssf.org/Compiler-Hardening-Guides/Compiler-Options-Hardening-Guide-for-C-and-C++.html

```
$ ./configure CXX=/usr/bin/clang++ PKG_CONFIG_PATH=/usr/local/lib/pkgconfig CXXFLAGS='-Wall
```

Note: Clang version 20.1.8 `-mbranch-protection=standard` results in exceptions not being caught.

On Debian 13 (Trixie) install the following libraries:

- libwxgtk3.2-dev
- libpugixml-dev

On Debian 12 (Bookworm) and Debian 13 (Trixie)

```
$ ./configure CXXFLAGS='-g -O0 -Wall -Wformat -Wformat=2 -Wimplicit-fallthrough -Werror=form
```

XPM Resources

The XPM plugin is not included in version 3.0.6 of Gimp for macOS.

- https://forums.wxwidgets.org/viewtopic.php?t=48030
- https://docs.gimp.org/3.0/en/gimp-scripting.html#gimp-plugins-install
- https://en.wikibooks.org/wiki/GIMP/Installing_Plugins

Use Gimp on Debian to export:

1. Change the image to be `Indexed` with `Menu > Image > Mode > Indexed...` and under `Generate optimum palette` set the `Maximum number of colours` to `8`
2. Scale the image with `Menu > Image > Scale Image...` and under `Image Size` select a width and height of `32`.
3. Export the image with `Menu > Export As...` providing filename with the extention of `xpm`.

Docs

The man page was created using help2man with the following command:

```
$ help2man --no-discard-stderr \
--name="A list manager for maintaining kit lists" \
--output=kitlist.1 ./kitlist
```

Optional Text-based User Interfaces (TUI)

Optionally there are two experimental build options for Kitlist providing Text-based User Interfaces.

FINAL CUT

> **Note:** Kitlist was developed with version `0.9.1` of FINAL CUT which requires patches from `./3rdparty/finalcut/patches/` to be applied to a custom build of that version. If FINAL CUT is installed with the `--prefix=/usr/local/`, kitlist will need to be configured with `PKG_CONFIG_PATH=/usr/local/lib/pkgconfig`, e.g.
>
> ```
> $ ./configure PKG_CONFIG_PATH=/usr/local/lib/pkgconfig --with-finalcut
> ```

Debian 13 (Trixie) includes version `0.9.1` of FINAL CUT which is likely to require the above mentioned patches. You can download and unpack the source from Debian with:

```
$ mkdir work
$ cd work
$ apt-get source libfinal-dev
```

On a Debian system, install FINAL CUT as follows:

1. Download and install the FINAL CUT library. **Note** when building FINAL CUT for Linux, to enable mouse support you need to also install the `gpm` library. If you need to install additional libraries to fix a failed build, re-run FINAL CUT's `configure` script so that it re-configures the build to use those libraries.

   - libgpm-dev
   - libncurses-dev

   E.g.

   ```
   $ sudo apt-get install libtool libgpm-dev libncurses-dev
   $ mkdir work
   $ cd work
   $ curl -L -o finalcut-0.9.1.tar.gz \
   https://github.com/gansm/finalcut/archive/refs/tags/0.9.1.tar.gz
   $ tar -xf finalcut-0.9.1.tar.gz
   $ cd finalcut-0.9.1
   $ for p in ../kitlist/3rdparty/finalcut/patches/0*.patch; do patch -p0 <$p; done
   $ autoreconf --install
   $ ./configure
   $ make
   $ sudo make install
   $ sudo ldconfig
   ```

On macOS, install the `ncurses` port from MacPorts. The `./configure` command will need to use the appropriate flags specifying the include directory for the for `ncurses` headers, otherwise `term.h` from macOS will be included first, producing errors similar to the following during the build phase:

```
/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs
__OSX_AVAILABLE(14.0) __IOS_AVAILABLE(17.0) __WATCHOS_AVAILABLE(10.0)
```

Where MacPorts has been installed under the `/opt/local` prefix, the `./configure` command should be:

```
$ ./configure CXX=/opt/local/bin/clang++-mp-20 CPPFLAGS=-I/opt/local/include
```

2. Include the `--with-finalcut` option when running `configure` for kitlist, e.g.:

```
$ ./configure --with-finalcut
$ make
```

3. Execute `kitlist` with the `--finalcut` option:

```
$ ./src/kitlist --finalcut
```

FTXUI

FTXUI provides another text-based front-end for Kitlist. After installing FTXUI, build Kitlist with `configure --with-ftxui`. It tends to crash or hang on Debian 13 (Trixie) on `aarch64` when compiled with GCC 14. Compiling both the application and the FTXUI library on the same platform with Clang 19 is stable.

The FTXUI version does not implement all the features of the GUI; in particular there is no facility to edit a category or an item name after they have been created. If unable to use another interface; the workaround is to edit the XML directly.

Building FTXUI from source.

See the CMake Wiki if you wish to specify a different compiler.

```
$ cd /usr/local/src/ftxui-6.1.9
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
```

Then build kitlist with:

```
$ cd /usr/local/src/kitlist
$ ./configure PKG_CONFIG_PATH=/usr/local/lib/pkgconfig --with-ftxui
```

```
$ make
$ ./src/kitlist --help
```