

lirc2xml: Extracting information from LIRC configuration files

Table of contents

1 Introduction.....	2
2 Usage.....	2
2.1 Limitation.....	2
3 Installation.....	3
4 Known bug.....	3
5 lirc2xml.....	3
6 Downloads.....	4

Date	Description
2010-03-20	Initial version.

Table 1: Revision history

1. Introduction

Converting LIRC configuration files to a raw representation is no easy task. The meaning of the different parameters are to some extent documented through comments in the code, but are not at all straightforward. I have therefore written `lirc2xml` as an extension to LIRC itself. It uses internal LIRC functions from `transmit.c` to "render" the signals. For this reason, it is available as a patch to the LIRC sources. No other form of distribution is planned.

The program generates the CCF (often called "Pronto format") representation of the raw signal, embedded in an XML file. Optionally, the DecodeIR library is invoked to attempt to decode the signals.

The program was [announced](#) on the Lirc mailing list on 2009-08-22. It did not receive any attention. Ideally, I would like to see the program in the future LIRC distribution, however I have a strong suspicion that LIRC-ers would rather see `foobar2lirc` than `lirc2foobar...`

The program runs from the command line in a Linux environment.

2. Usage

The usage is given by:

```
Usage: lirc2xml [options] [config-file]
       -h --help                display this message
       -v --version             display version
       -o --output=filename     XML output filename
       -r --remote=remotename  Include only this remote in the
export                             export
       -d[debug_level] --debug[=debug_level]
```

The Lirc configuration file, defaulting to the system's configuration file, is read and translating into an output file, defaulting to `/tmp/basename-of-configfile.xml`. Per default, all remotes in the configuration file are translated to the output file, unless, using the `-r/--remote`-option is given, in which case only the selected remote is processed. If configured, John Fine's library DecodeIR is invoked for each signal. It will try to identify the signal as one of a number of known IR signal protocol.

2.1. Limitation

The present version does not render toggle signals (more precisely: it renders them only

for one value of the toggle). Also, the treatment of repeats may be incorrect.

3. Installation

The following instruction is intended for users with some experience with installation programs under Linux.

1. (Optional, but highly recommended.) Install `libDecodeIR.so` to a directory like `/usr/local/lib`, where the linker will find it.
2. Download and unpack `lirc-0.8.6`. Probably other, similar, versions will also work.
3. Change to the top of the recently unpacked tree and apply the patch by `patch -p1 < path-to-lirc2xml.patch`.
4. Regenerate some autoconfig files by `autoreconf -f -i -s`.
5. Configure by, e.g., `./configure --with-driver=none --enable-decodeir`
6. The command `make` will now, in addition to all the usual Lirc programs, create the `lirc2xml` program, located in the `tools` subdirectory.
7. `make install` will install it together with the rest of the programs in the Lirc package, per default in `/usr/local/bin`. Actually, just copying `tools/lirc2xml` should (in this case) suffice.

4. Known bug

Command names in Lirc files containing less-than characters (<) and ampersands (&) will generate nonvalid XML. I have no intention to fix this in the short time span. It is hard to write waterproof code for that without using xml libraries; that kind of characters in command names is silly anyhow, and broken XML it can also easily be fixed with a text editor.

5. lirc2xml

Mainly as a proof of concept, I wrote a simple back-end to the `lirc2xml` program: `lirc2rmdu`, written in [Python](#). (Possibly some would argue, that it should be called `xml2rmdu`, however I felt that would be more misleading.) It transforms the xml file from `lirc2xml` to a `rmdu`-file to be used by [RemoteMaster](#) program to design a "device upgrade" in the JP1 context. Note that, for several different reasons, the output of the program is not meant to be a perfect `rmdu` file, but rather a first starting point, however saving a lot of work transforming what can be automatically transformed. In particular, the device number and `-parameters` need some fixing. possibly with a text editor outside of `RemoteMaster` -- actually I manage to "hang" the program on a few occasions. Fixing this may be possibly by parsing `RemoteMaster's protocols.ini` file, but I am not sure that is a wise way of spending my time...

```
Usage:
  lirc2rmdu.py [-r remote] <xmlfile>
```

The program will extract the remote given by the `-r` argument (or the first one if not

given), and write it to a file in the current directory, using a named taken from the remote's name.

6. Downloads

- [lirc2xml.patch](#)
- [lirc2rmdu](#)
- [lirc-0.8.6-gz](#). Also "sufficiently similar" versions will do.
- DecodeIR version 2.40 (other versions will also work); either the [sources](#), or the [precompiled package](#) (which includes 32- and 64-bit Linux versions).