Using RedRat3 on Linux

Introduction

This guide is intended to explain how to use RedRat3 hardware on Linux. It is assumed that the reader is familiar with Linux systems, capable of installing software on them and comfortable executing commands from a terminal.

Most of the tools described herein were not written by us, so this information may be inaccurate, out of date, or just plain wrong. We encourage the reader to verify that our examples are correct by reading the man pages before use.

Command examples are written in *italics*.

Prerequisites

Recent kernel

Using the RedRat3 requires the RedRat3 driver to be installed. This has been included in the Linux kernel since version 3.3, so if you have a more recent kernel than this you're all set. We maintain instructions for older kernels on our website.

ir-keytable

This program is "a swiss-knife tool to handle Remote Controllers". It can be used to verify device functionality and to adjust various settings for decoding signals and mapping remote control key presses to events.

Verifying that the RedRat3 has been detected

Run *ir-keytable* to verify that the RedRat3 has been found and that the correct driver has been loaded. The response should look something like this:

```
Found /sys/class/rc/rc0/ (/dev/input/event6) with:
      Driver redrat3, table rc-hauppauge
```

The exact text doesn't matter, so long as it found something with a redrat3 driver. If that didn't work, check the output of the kernel message log with *dmesg*. Also make sure that the RedRat3 is actually plugged in!

Sending and receiving signals with LIRC

Installation

The easiest way to install LIRC is to use the package manager or software repository for your distribution. If you are prompted to select a driver to use following the installation, pick the option of "none", "other" or something similar. As mentioned earlier, the driver for the RedRat3 is included in the kernel so it is not necessary to configure LIRC to use a special driver.

We neither recommend nor support building LIRC from source code.

Architecture

Most of the tools that come with LIRC communicate with lircd, the LIRC daemon. lircd has sole access to the hardware and is also responsible for decoding IR signals. This architecture allows multiple tools to be run at the same time by normal users.

Device files

Each RedRat3 will have a corresponding device file under /dev. The naming convention varies slightly between distributions, but the default name for a single device is usually /dev/lirc or /dev/lirc0. Run *ls /dev/lirc** to see which device(s) you have and make note. If you have none, check the kernel message logs (run *dmesg*) to see what went wrong. Also double check that the RedRat3 really is plugged in.

Starting and stopping lircd

If you installed LIRC through a package manager it probably set up a system service or boot script to start lircd automatically. To verify that lircd is running, run *pidof lircd* in a terminal. If it returns a number, lircd is running.

Note that running more than one instance of lircd at the same time is not allowed because it needs dedicated access to the hardware. Attempting to start a second instance of lircd will produce an error message.

To stop lircd you can either stop the service, if there is one, or do *killall lircd* as root.

To start lircd as a background process, simply run *lircd* as root. To have lircd use a specific device, e.g. /dev/lirc0, run it with *lircd –d /dev/lirc0* . To run it as a foreground process, allowing you to view status messages, add the *–n* switch.

Creating a configuration file

LIRC requires a configuration file to be able to transmit or receive IR signals. This file essentially maps IR signals to button events. The easiest way to get one is to just download it. A list can be found here http://lirc.sourceforge.net/remotes/ , and even if you can't find one that works perfectly, it's a good place to start.

LIRC comes with a tool called irrecord for generating configuration files. irrecord is unlike the other tools in that it accesses the hardware directly, so it cannot be used while lircd is running. Run it as root with *irrecord my-config-file* to have it write a configuration file to my-config-file. As with lircd you can specify which device to use with *–d* . irrecord outputs lots of helpful text as it runs, so just follow its instructions.

There are times when irrecord simply doesn't work, or it claims to have worked but the resulting configuration file isn't right. If this is the case, you might try the Signal Database Utility available from our website. This is only available on Windows, but it allows signal capture and export to the Linux LIRC format. Usage is documented on our website.

Testing a configuration file

Once you have a configuration file it's a good idea to see if it actually works! lircd accepts the path to a configuration file as an argument, so you can run lircd manually with *lircd –n my-config-file* to see what happens. The *–n* switch prevents lircd from becoming a daemon so you can read its output in the terminal. Make sure there are no errors listed here, particularly anything about a bad configuration file.

Run *irw* in another terminal to output every signal lircd successfully decoded. Also keep an eye on the output from lircd. Each button press on your remote should produce one or more lines from irw. If it doesn't, you may need to edit the tolerances in the configuration file (aeps and eps), or maybe go back a step and generate a new configuration file.

Updating the LIRC configuration file

The default location for the LIRC configuration file is /etc/lirc/lircd.conf . You can either overwrite this file with your new configuration file merge the two together by appending the contents of your new file. Merging is useful if you've just recorded a new remote but don't want to lose your existing ones. Editing /etc/lirc/lircd.conf will require root access.

Once you've edited lircd.conf you'll probably need to restart lircd for the changes to take effect.

Listing signals

If you ever forget which remotes and signals have been configured, or you want to check that your new configuration file has been loaded properly, run *irsend LIST "" ""* to print the list of remotes in the configuration file. You can also use *irsend LIST remote-name ""* to list the signals configured for a given remote.

Transmitting signals

irsend can be used to transmit signals. To send a signal once, use *irsend SEND_ONCE remote-name signal-name*. It is also possible to send multiple signals consecutively or to repeat signals until stopped – see the man page for details.

If running this command seems to have no effect, check that the RedRat3 actually transmitted a signal. The easiest way to do this is to view the front of the device with a mobile phone camera, webcam or similar device. IR signals should appear as bright flashes (unless your camera has excellent IR filtering). There is a red LED inside the device that blinks when transmitting, though this can be difficult to see through the black casing.

If you find no evidence that the RedRat3 transmitted anything then something has gone wrong – check the kernel message logs and try running lircd with *–n* so you can observe any error messages it may produce. If the RedRat3 appears to be transmitting but the signals are not being recognised, your configuration file may not be quite right. Try creating it again as described previously. Alternatively, if you have another RedRat3, record the transmitted signal with our Signal Database Utility and see if it matches what you expect.