Razer device configuration tool

https://bues.ch/h/razercfg

This is a configuration utility for Razer devices on Linux systems.

Supported devices

Device support table at https://bues.ch/h/razercfg#device_support

Dependencies

- Python 3.x: https://www.python.org/
  Debian Linux: `apt-get install python3`
- libusb 1.0: http://libusb.org/
  Debian Linux: `apt-get install libusb-1.0-0-dev`
- PySide (for the graphical qrazercfg tool only): https://wiki.qt.io/PySide
  Debian Linux: `apt-get install python3-pyside`
- cmake 2.4 or later (for building only): https://cmake.org/
  Debian Linux: `apt-get install cmake`

Note that almost all distributions ship prebuilt packages of the above dependencies.

If you installed a dependency after you already ran `cmake .` and/or `make`, it might happen that the dependency is still not found. Just delete the cmake status files or unpack a clean razercfg tarball to workaround this issue.

Building

First invoke `cmake` to build the makefiles. Then invoke `make` to build the binaries:

```
cmake .
make
```

(Note the required space and dot after the cmake command)

Installing

First you need to install the tool libraries and binaries. Do this by executing the following command as root:

```
make install
```

Be aware that `make install` installs the shared library `librazer.so` to `$PREFIX/lib`. The default `$PREFIX` is `/usr/local/`, but the install prefix can also be changed via `-DCMAKE_INSTALL_PREFIX='<somewhere>'`. You have to make sure that `librazer.so` in `$PREFIX/lib/` can be found by the dynamic linker `ld.so`. Your operating system most likely already has support for libraries in `/usr/local/lib`. So on most systems you don't have to do anything. If this is not the case, or you installed razercfg somewhere else, a new library search path can be added via `/etc/ld.so.conf` or `/etc/ld.so.conf.d/`. See your operating system manual for further information.

If you use systemd:

The `make install` step installed the razerd.service file. Reboot or run the following command as root to start the razerd daemon:

```
systemctl start razerd
```

If you do not use systemd:

To automatically start the required system daemon `razerd` at bootup time, you need to install the init-script. This software package includes a generic example script, that should work out-of-the-box on many Linux distributions. To install it, invoke the following commands as root:

```
cp ./razerd.initscript /etc/init.d/razerd
ln -s /etc/init.d/razerd /etc/rc2.d/S99razerd
ln -s /etc/init.d/razerd /etc/rc5.d/S99razerd
ln -s /etc/init.d/razerd /etc/rc0.d/K01razerd
ln -s /etc/init.d/razerd /etc/rc6.d/K01razerd
```

If you use udev:

The `make install` step installed the udev script to

```
$(pkg-config --variable=udevdir udev)/rules.d/80-razer.rules
```

This should work on most distributions.

If udev notification does not work, try to reboot the system.

RazerD Configuration

The user may create a razerd configuration file in `/etc/razer.conf` which can be used to specify various razerd options and initial hardware configuration settings. An example config file is included as `razer.conf` in this package. If no configuration file is available, razerd will work with default settings.

X Window System (X.ORG) Configuration

If you don't have an xorg.conf, you don't have to do anything and it should work out-of-the-box.

X must *not* be configured to a specific mouse device like `/dev/input/mouse0`. On configuration events, razerd may have to temporarily unregister the mouse from the system. This will confuse X, if it's configured to a specific device. Configure it to the generic `/dev/input/mice` device instead. This will enable X to pick up the mouse again after a configuration event from razerd.

Example xorg.conf snippet:

```
Section "InputDevice"
    Identifier  "Mouse"
    Driver  "mouse"
    Option  "Device" "/dev/input/mice"
EndSection
```

Alternatively, do not specify a `"Device"` at all. X will autodetect the device then:

```
Section "InputDevice"
    Identifier  "Mouse"
    Driver  "mouse"
EndSection
```

In any case, do *NOT* use: `Option "Device" "/dev/input/mouseX"`

Using the tools

To use the tools, the razerd daemon needs to be started as root, first. Without the background daemon, nothing will work. The daemon is responsible for doing the lowlevel hardware accesses and for tracking the current state of the device. While the daemon is running, the user interfaces `razercfg` (commandline) and `qrazercfg` (graphical user interface) can be used.

Uninstalling

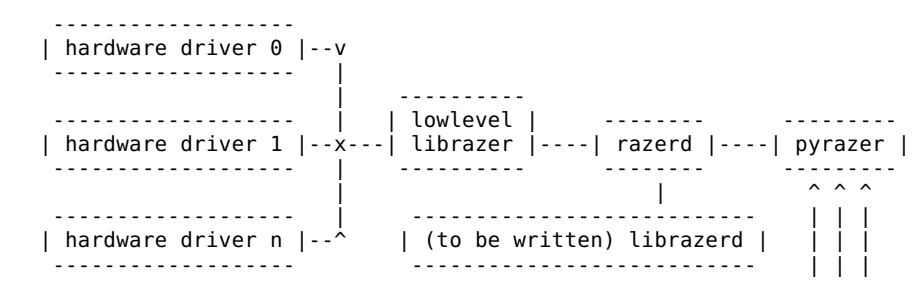
If you installed razercfg with your distribution packaging system, use that to uninstall razercfg.

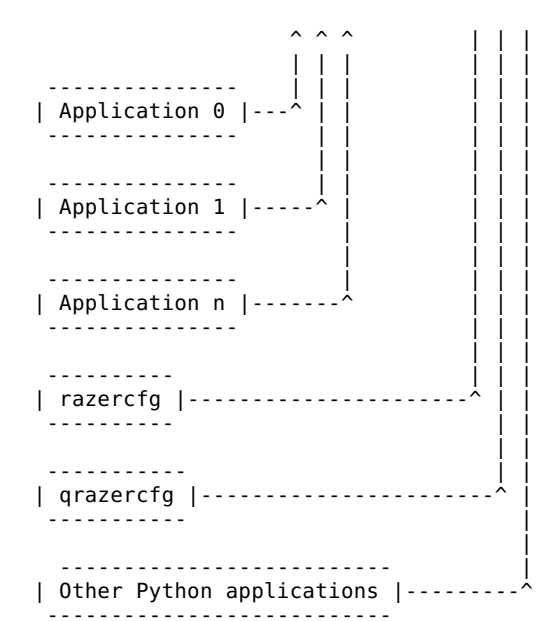
If you compiled razercfg from source and installed it with `make install`, you can use the `uninstall.sh` script from the razercfg archive to uninstall razercfg from the system. It must be called with the install prefix as its first argument. That usually is `/usr/local`, unless specified otherwise in cmake. A call to uninstall.sh might look like this:

```
./uninstall.sh /usr/local
```

Architecture

The architecture layout of the razer tools looks like this:





So in general, your application wants to access the razer devices through pyrazer or (if it's not a python app) through librazerd. (Note that librazerd is not written, yet. So currently the only way to access the devices is through pyrazer). Applications should never poke with lowlevel librazer directly, because there will be no instance that keeps track of the device state and permissions and concurrency.

License

Copyright (c) 2007-2016 Michael Büsch, et al.

See the COPYING file for license information.