Linux powered coffee roasting

Andrew Tridgell
Samba Team
Coffee Roasting

• Inputs
  – green coffee beans
  – heat (needs to get to about 210°C)
  – stirring (to distribute the heat)
  – time (about 15 minutes)

• Outputs
  – roasted coffee!

• Rule of 2
  – green beans OK for 2 years
  – roasted beans OK for 2 weeks
  – ground coffee OK for 2 minutes
Corretto roaster

- Lots of coffee roasting communities
  - coffeesnobs.com.au is best known one in Australia
  - very active group of enthusiasts
- 2006 - “Corretto” roaster started
  - bread machine
  - heat gun
  - listen to the roast!
pyRoast

- Python coffee roasting control
  - feedback loop for temperature control
  - configurable roast profiles
  - data logging

- Temperature control support
  - USB thermocouple
  - 'paulus' thermocouple

- PyQt4 interface
  - built with designer-qt4
Power control

• How to control power of a heat gun?
  – needs to be controllable from pyRoast
  – solution: ask Paulus!
USB protocol analysis

• How to decode a USB protocol?
  – have a windows driver, no Linux driver

• Usual setup …. 

![Diagram of USB protocol analysis with nodes for Windows Driver, USB device, and USB sniffer connected by lines.](attachment:diagram.png)
protocol filtering

- Much easier if you can modify packets
  - can test protocol format theories
  - fast development cycle
VM USB filtering

- **LD_PRELOAD** intercept
  - preload reads instructions from a file
  - developer uses text editor to control USB filtering
USB intercept

- Userspace USB controlled by ioctl()
  - USBDEVFS_SUBMITURB
    • send a USB USB to a device
  - USBDEVFS_REAPURBNDELAY
    • receive reply from device

- Preload hooks
  - open() - check if device name matches, remember fd
  - ioctl() - intercept REAPURB, possibly replace data

- Simple replacement
  - check USB data size
  - replace data with hex bytes from /tmp/usb.data
Protocol analysis

• Stage 1: experimenting
  – start with an existing data blob
  – edit and watch result on windows display
  – try to work out patterns

• Stage 2: theory testing
  – write tool to produce desired data
  – test against windows display

• Stage 3: write Linux driver
  – easy once you know the protocol
Questions?

- More info:
  - Coffeesnobs:
  - usb_preload:
    - http://git.samba.org/?p=tridge/junkcode.git;a=tree;f=preload_usb
  - DMM app
    - http://git.samba.org/?p=tridge/junkcode.git;a=tree;f=DMM