Dynamic Device Handling on the Modern Desktop

David Zeuthen <davidz@redhat.com>
Kay Sievers <kay.sievers@suse.de>

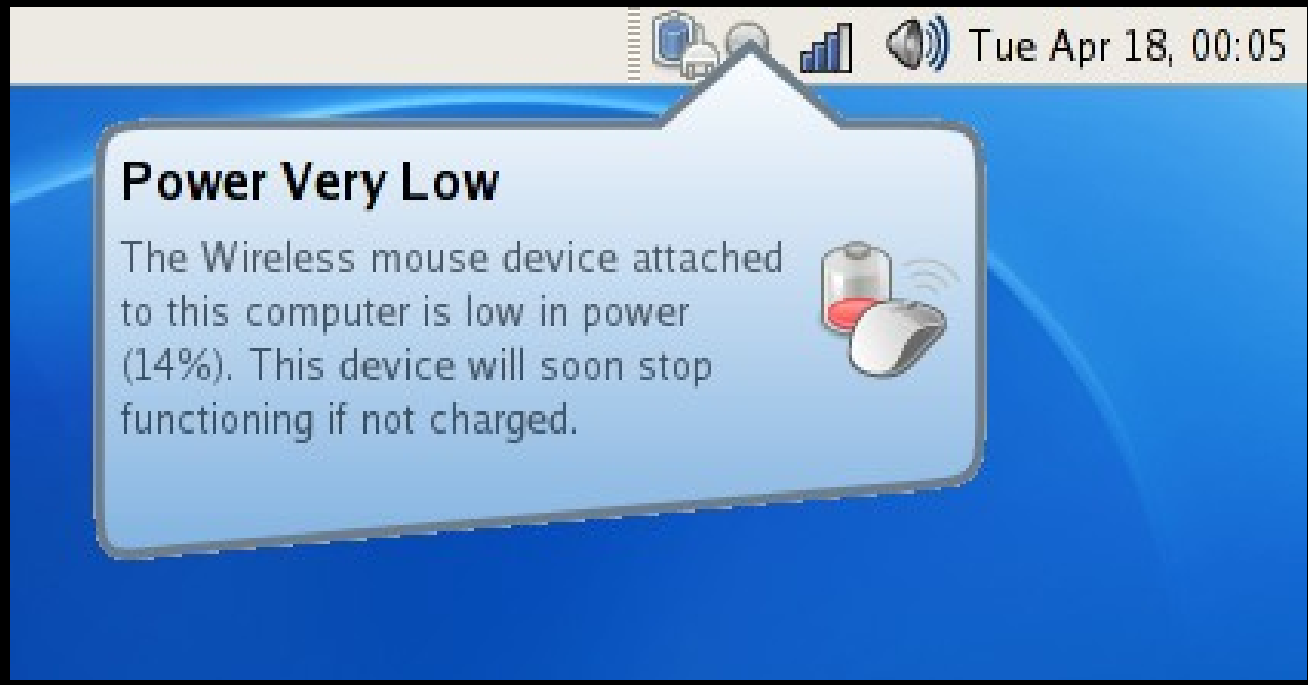
Linux Symposium
Ottawa, Canada
July 2006

Example

- Plugging in an a digital camera

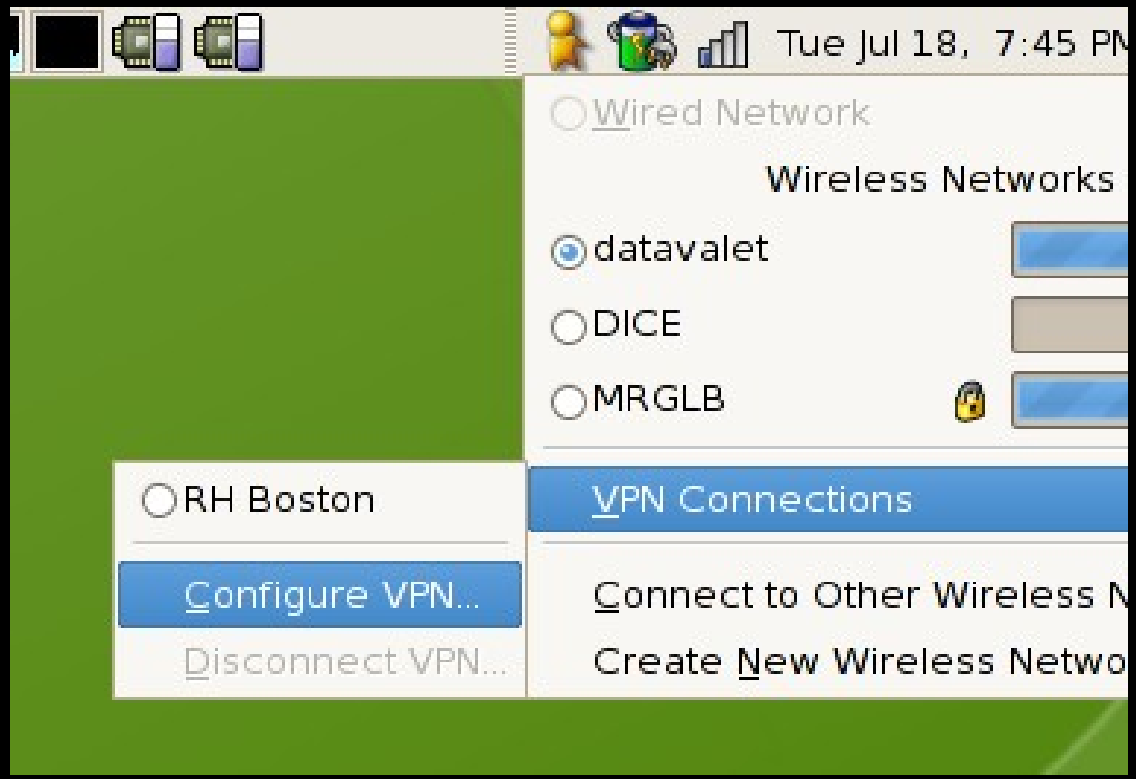
  user space driver (libgphoto2) provides XML files for identifying hardware

Example



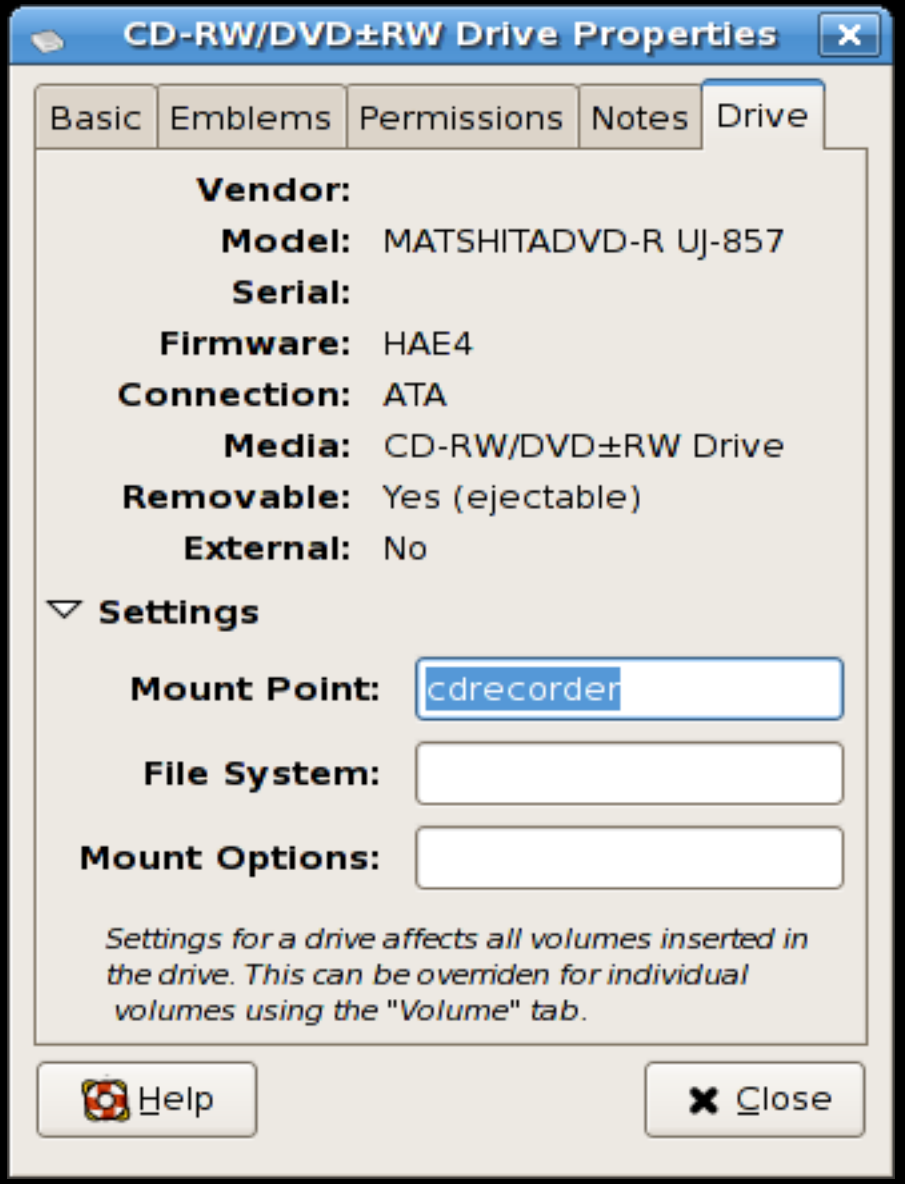
- Integrated Power Management
  - Support for: ACPI, APM, PMU, USB UPS's, wireless USB keyboard/mice batteries

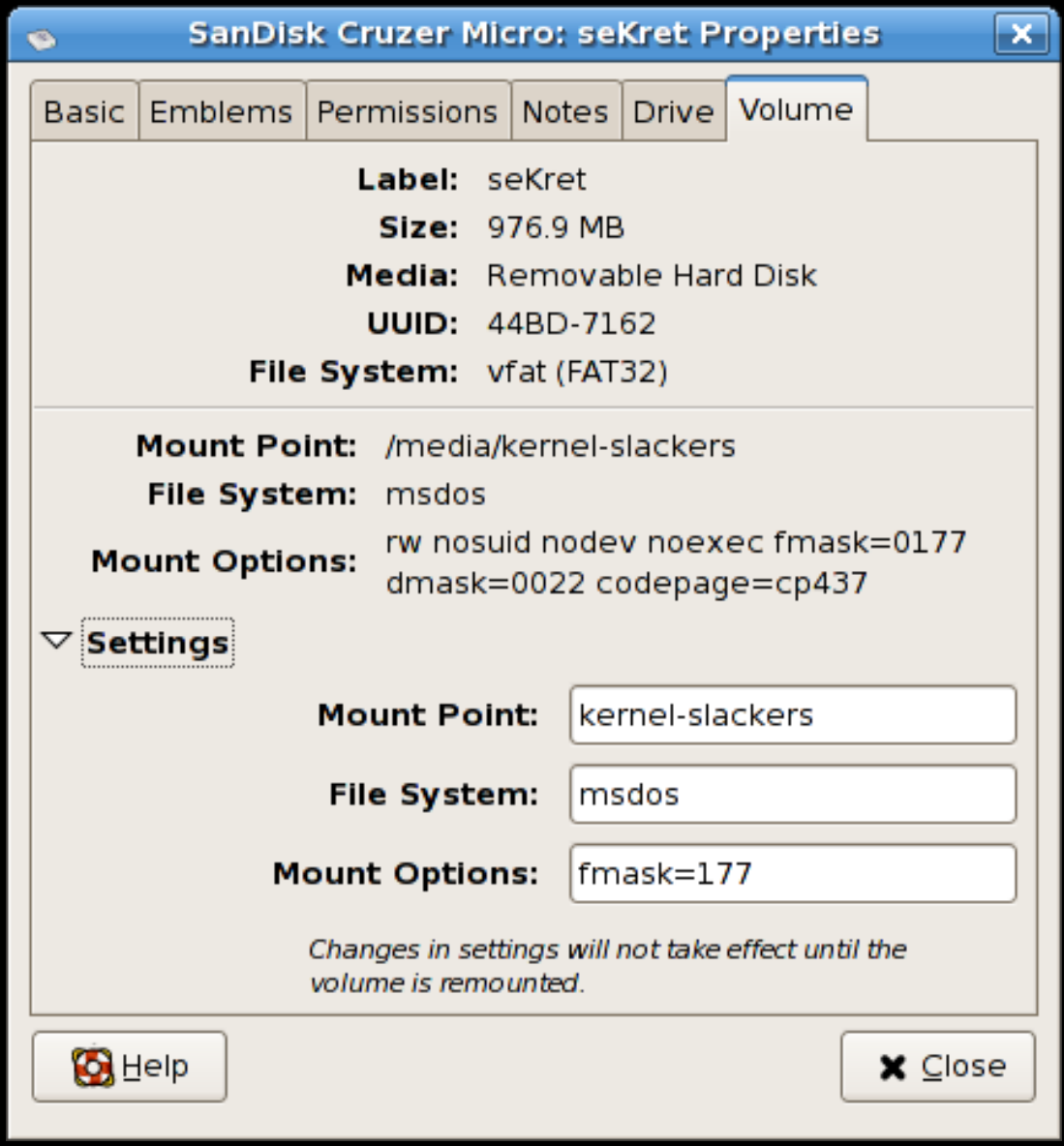
Example



- Network Management
- Wired, Wireless, Ad-Hoc
  - WEP, WPA – passwords stored in user's keyring
  - plug-in based VPN (vpnc, openvpn, ipsec, pptp)

Example





mount preferences stored in user's desktop session - per drive and/or volume

Demo

- Plugging in an USB stick with an encrypted file system

  LUKS, dm-crypt, cryptsetup

How does all this work?

- kernel
- udev
- HAL
- D-BUS
- User session

How does all this work?

- The kernel discovers the device and exports its state in sysfs

```
/sys/block/sda
|-- dev
|   |-- iosched
|   |   |-- antic_expire
|   |   |-- est_time
|   |   |-- read_batch_expire
|   |   |-- read_expire
|   |   |-- write_batch_expire
|   |   `-- write_expire
   ...
|   |-- read_ahead_kb
|   `-- scheduler
|-- range
|-- removable
|-- sda1
|   |-- dev
|   |-- size
|   |-- start
|   `-- stat
...
```

How does all this work?

- udev receives the events over netlink

```
add@/class/input/devices/input5
ACTION=add
DEVPATH=/class/input/devices/input5
SUBSYSTEM=input
SEQNUM=1166
...
```

```
recv(3,"add@/class/input/devices/input5\0ACTION=add\0DEVPATH=/class/
input/devices/input5\0SUBSYSTEM=input\0SEQNUM=1166\0CLASS=/class/inp
ut/devices\0PHYSDEVPATH=/devices/pci0000:00/0000:00:1d.1/usb2/2-2/2-
2:1.0\0PHYSDEVBUS=usb\0PHYSDEVDRIVER=usbhid\0PRODUCT=3/46d/c03e/2000
\0NAME=\"Logitech USB-PS/2 Optical Mouse\"\0PHYS=\"usb-0000:00:1d.1-
2/input0\"\0UNIQ=\"\"\0EV=7\0KEY=70000 0 0 0 0 0 0 0 0\0REL=103\0",
2048, 0) = 383
```

How does all this work?

- udev creates device nodes, runs modprobe and other programs, and passes the event over a socket to the HAL daemon

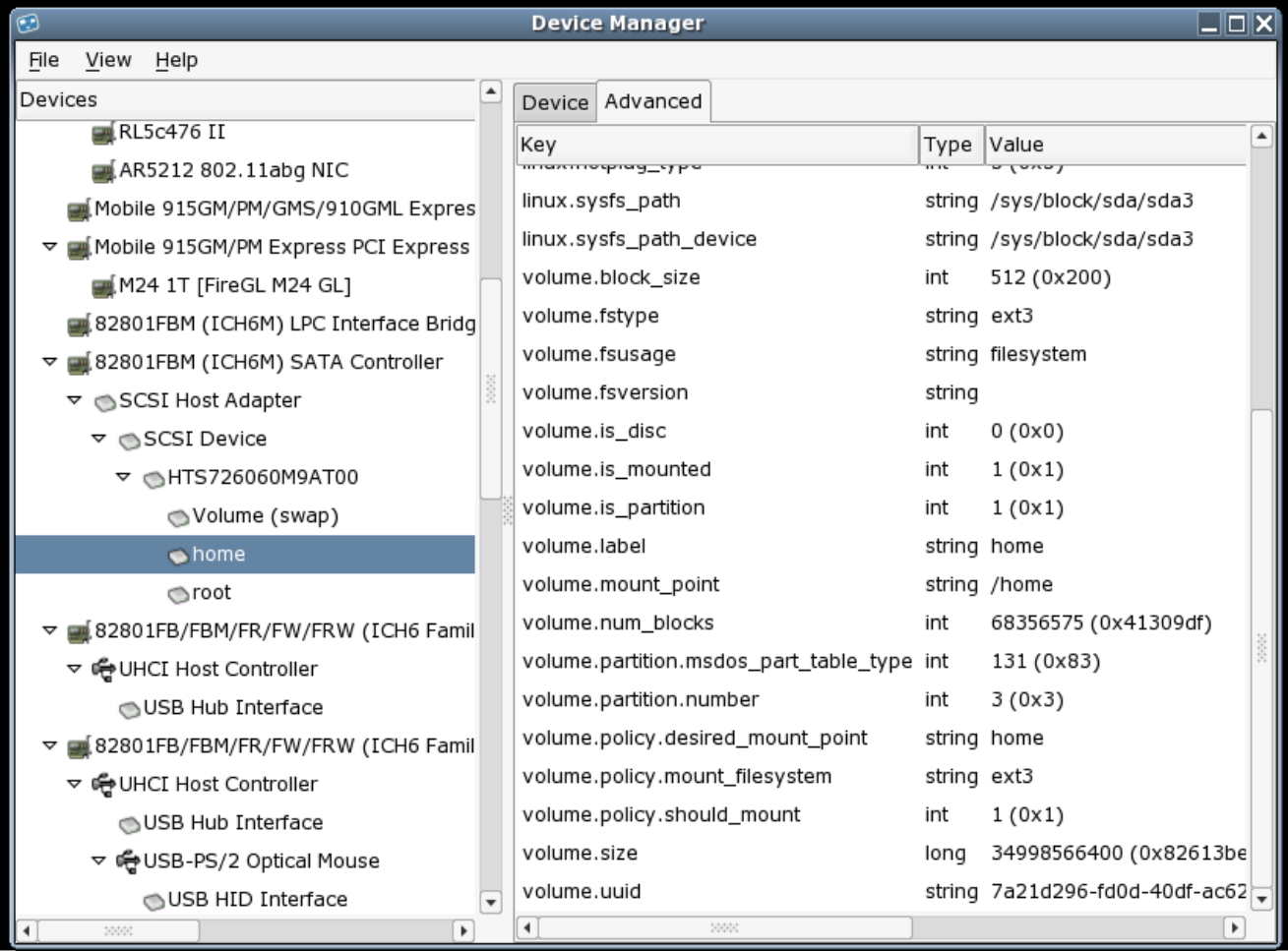
```
KERNEL=="mice",  NAME="input/$kernel_name", MODE="0640"
ACTION=="add", SUBSYSTEM=="pci", MODALIAS=="?*",
RUN+="/sbin/modprobe $modalias"
SUBSYSTEM=="scsi_device", ACTION=="add", RUN+="/sbin/modprobe sg"

IMPORT{program}="/sbin/vol_id --export $tempnode" \
  ENV{ID_FS_UUID}=="?*", SYMLINK+="disk/by-uuid/$env{ID_FS_UUID}"

RUN+="socket:/org/freedesktop/hal/udev_event"
```

How does all this work?

- HAL receives the event, investigates the device, and creates a representation in its own device tree



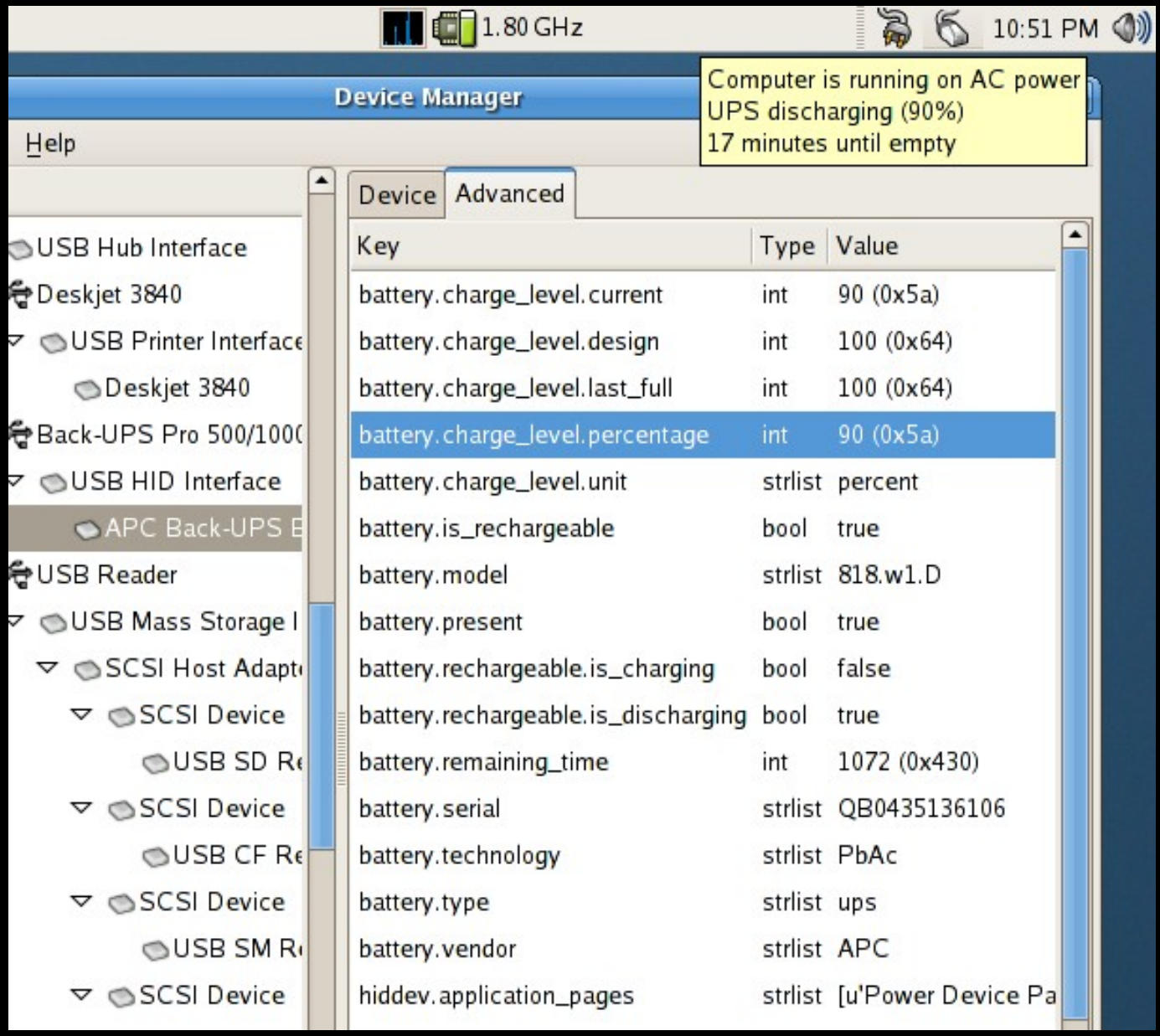
How does all this work?

- HAL merges properties into the device object from
  - what the kernel knows
  - poking the device
  - the system configuration
  - hardware databases
    - quirks
    - information
    - user space drivers

How does all this work?

- HAL *addons*
  - Kind of a “daemon” process tied to a device
  - HAL controls the addon's life cycle
  - Sort of a *“Driver”* in user space:
    - usbhid UPS
    - Backlight LCD controls
    - Battery levels for wireless mice and keyboard
  - ... but also used to e.g. poll removable storage drives every two seconds

How does all this work?

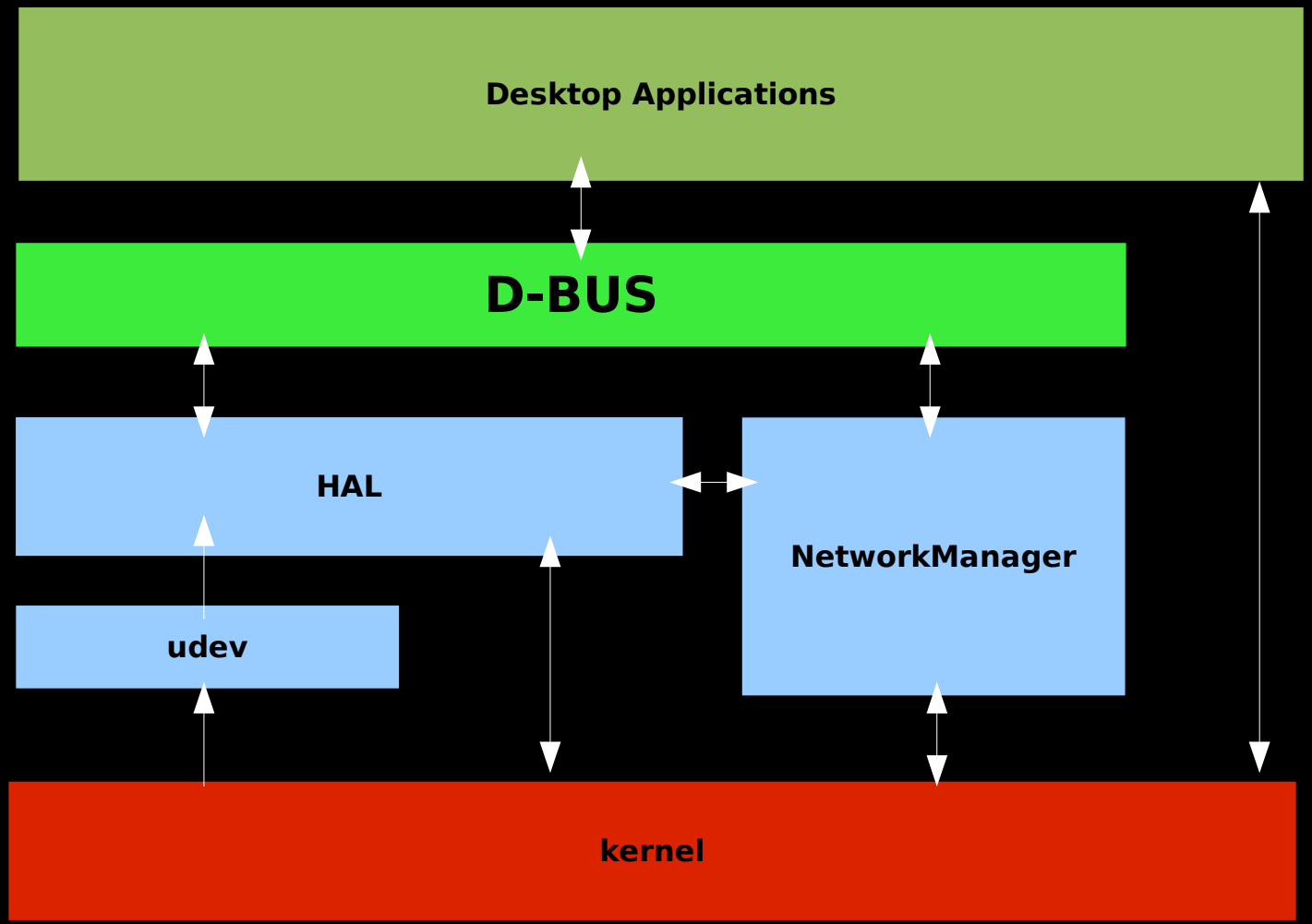


- For the UPS, the battery.* properties are populated from the HAL *addon*

How does all this work?

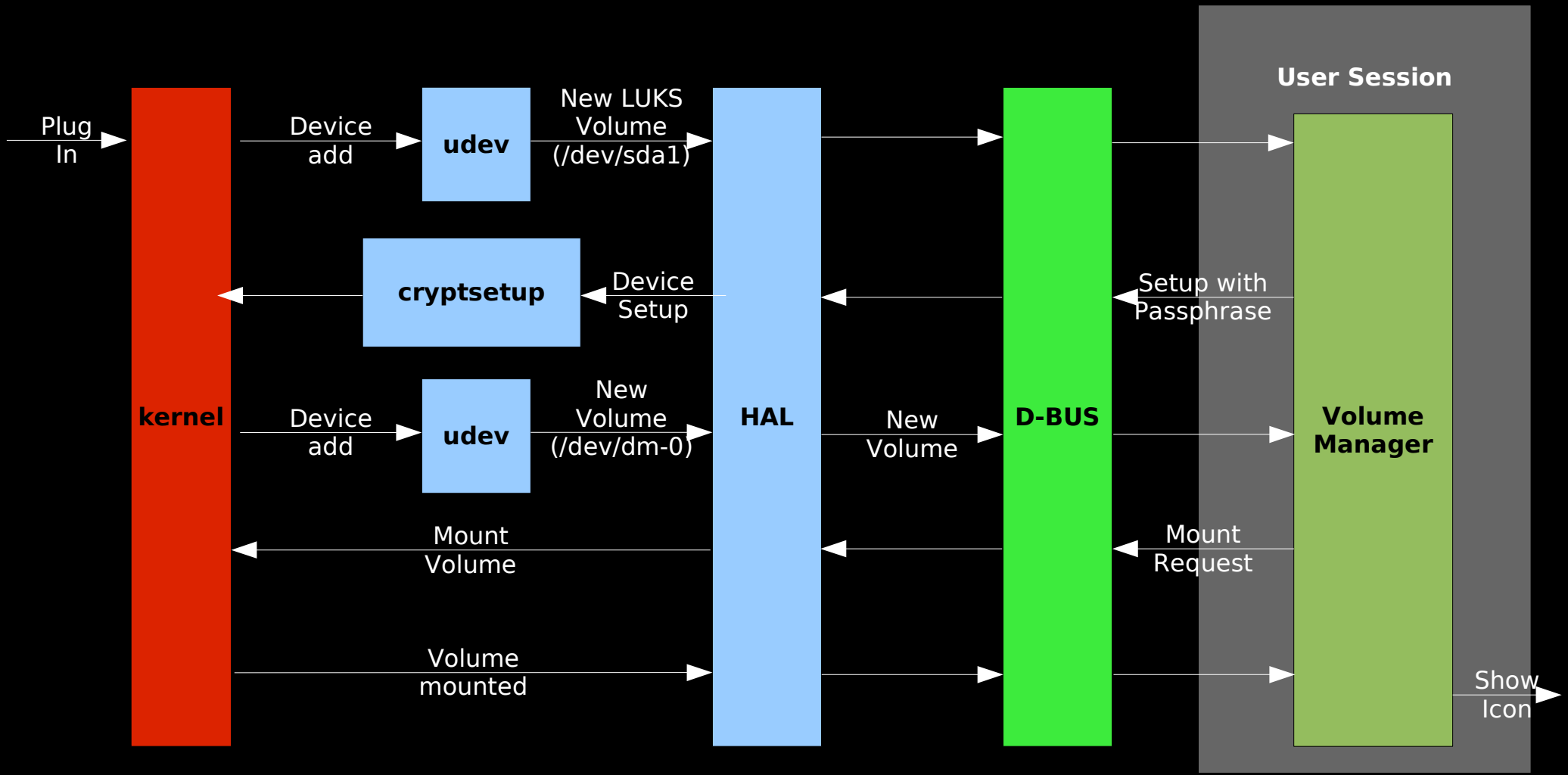
- HAL uses D-BUS to tell all subscribed listeners about the new device
- The user session reacts
  - starts programs
  - configures device according to user settings
  - shows icons
  - running programs update their user interface
  - ...

How does all this work?



How does all this work?

- LUKS encrypted storage setup



Justification

- Moves device / system – specific knowledge from lots of applications into a central location
  - Also beats the crap out of your drivers, e.g. NetworkManager
- Same device discovery / configuration API on all Linux distributions and other OS'es
  - Encourages to store settings in the user session rather than in /etc
- **Allows GNOME, KDE to actually sanely integrate with the system**

What's next?

- Continue providing useful API's for desktop use, for example
  - Bluetooth
  - Disk formatting
- Performance / Optimization work
  - Not a problem on the usual desktop PC
  - Important for
    - Embedded (Nokia Maemo, OLPC, Palm etc.)
    - Big Iron (thousands of devices)
  - We haven't done much work on this yet

What's next? (cont'd)

- Privileges - “what users are allowed to use the API's in HAL”
  - Right now it's all or nothing (console user or group membership). Broken.
  - Working on a freedesktop.org project to address this (PolicyKit)
- 1.0?
  - API/ABI stability - for HAL this means stability of the API's provided via D-BUS
  - API stability of HAL extension points?
  - Needs a lot of thought – and work...

Call to Action

- Drivers
  - Make them robust enough to handle poking by HAL and NetworkManager
  - Export useful information in sysfs
- Be a good citizen
  - Don't ignore hotplug / udev
    - We're looking at you device-mapper
- Error reporting other than *printk* please
  - `Jun 18 19:01:54 zelda kernel: NTFS-fs error (device sda4): load_system_files(): Volume is dirty.  Mounting read-only.  Run chkdsk and mount in Windows.`

Call to Action

- HAL authors are not kernel experts, yet it touches a lot of subsystems
  - Look at HAL sources to see if HAL does anything stupid with your subsystem
- Don't write stand-alone user space daemons that read settings from /etc
  - It just doesn't integrate well with the desktop – settings need to come from the user session
- Start participating
  - Mailing list: hal@freedesktop.org
  - IRC: #hal on freenode