AUUG 2004 Conference
Who Are You?

1 September 2004

Using Vital Product Data For Persistent Device Naming

Martin Schwenke
IBM OzLabs Linux Technology Center
<martins@au.ibm.com> · <martin@meltin.net>







The plan. . .

- Introduction.
  - Linux®device naming history and state-of-the-art.
  - Vital Product Data (VPD) and hardware inventory.
- My proposal.
  - Basics.
  - Device naming scheme #1.
  - Device naming scheme #2.
  - Device naming using VPD.
- Progress.
  - Does it work?
  - Will it work?
- Conclusions, questions, . . .







History

- Device names derived from major/minor numbers.







History

- Device names derived from major/minor numbers.
- Standard names: LANANA, FHS.







History

- Device names derived from major/minor numbers.
- Standard names: LANANA, FHS.
- Linux 2.4 `devfs`.







History

- Device names derived from major/minor numbers.
- Standard names: LANANA, FHS.
- Linux 2.4 `devfs`.
- Linux 2.6 `udev`.







udev

- Gets Linux 2.6 `sysfs` path via `hotplug`.
- Rule files use `sysfs` attributes.
- Generally, VPD not (yet?) in `sysfs`.
- Has call-out facility.
- Combine with `scsi_id`:

```
BUS="scsi", PROGRAM="/sbin/scsi_id", RESULT="0123456789", NAME="yip"
```







udev

- Gets Linux 2.6 `sysfs` path via `hotplug`.
- Rule files use `sysfs` attributes.
- Generally, VPD not (yet?) in `sysfs`.
- Has call-out facility.
- Combine with `scsi_id`:

```
BUS="scsi", PROGRAM="/sbin/scsi_id", RESULT="0123456789", NAME="yip"
```

- Want to generalise this. . .







Vital Product Data

```
*DS IDE Disk Drive
*AX /dev/hda
*MF Hitachi
*TM HTS548080M9AT00
*SN MRL422L4GE3VDB
*RM MG4OA5BA
*YL 0000:00:07.1/0.0
```

VPD and Hardware Inventory

- WWID (Z7)
- MF+TM+SN
- YL







VPD and Hardware Inventory

- WWID (Z7)
- MF+TM+SN
- YL

- `lspci`?
- `kudzu` and `hwinfo`?
- WBEM/CIM?
- OpenHPI?
- HAL?
- lsvpd







Proposal Basics 1

1. Retrieve device VPD.
2. Name device.
3. Plug name into VPD.







Proposal Basics 2

- Single call-out for all naming?

```
PROGRAM="/sbin/namedev %k %n %b %M %m", NAME="%c{1}", SYMLINK="%c{2+}"
```

- .d-style directories of “namers”.







Proposal Basics 2

- Single call-out for all naming?

```
PROGRAM="/sbin/namedev %k %n %b %M %m", NAME="%c{1}", SYMLINK="%c{2+}"
```

- .d-style directories of “namers”.
- Namers may access VPD database.







#1: First Name Wins

```
#!/bin/sh
for i in /etc/namedev/namers.d/* ; do
    if [ -x "$i" ] ; then
        names=$($i "$@")
        if [ -n "$names" ] ; then
            echo "$names"
            break
        fi
    fi
done
```







#1: First Name Wins

```
#!/bin/sh
for i in /etc/namedev/namers.d/* ; do
    if [ -x "$i" ] ; then
        names=$($i "$@")
        if [ -n "$names" ] ; then
            echo "$names"
            break
        fi
    fi
done

/etc/namedev/namers.d/99default:
#!/bin/sh
echo "$1"  # kernel name
```







#2: Each Namer Can Override

```
#!/bin/sh
for i in /etc/namedev/namers.d/* ; do
    if [ -x "$i" ]  ; then
        names=$($i $names)
    fi
done
```







#2: Each Namer Can Override

```
#!/bin/sh
for i in /etc/namedev/namers.d/* ; do
    if [ -x "$i" ]  ; then
        names=$($i $names)
    fi
done

/etc/namedev/namers.d/00default:
#!/bin/sh
echo "$1"  # kernel name
```







Pros and Cons?

- Approach #2 is more flexible than #1.
- Persistence is troublesome...
- ...it needs to be a 2 pass process.
- For simplicity, use 2 pass version of #1 for now...







Device Naming Using VPD

- Separate VPD-based namers look at different VPD elements?







Device Naming Using VPD

- Separate VPD-based namers look at different VPD elements?
- No!







Device Naming Using VPD

- Separate VPD-based namers look at different VPD elements?
- No!
- Let's encode the rules into the "database" structure...







Database Initialisation

```
mkdir /etc/namedev/vpdnamer.d
cd /etc/namedev/vpdnamer.d
mkdir 00-sd-Z7
mkdir 10-sd-MF,TM,SN
mkdir 20-sd-MF,TM
```

vpdnamer

/etc/namedev/namers.d/50vpdnamer:

```
for d in /etc/namedev/vpdnamer.d/* ; do
  [ -d "$d" ] || continue
  rule="${d#/*/*-}"
  type="${rule%-*}"
  case "$1" in
    ${type}*)
      ks=$(echo "${rule#*-}" | sed -e 's/,/ /g')
      lookup "$d" $ks
      ;;
  esac
done
```

vpdnamer - lookup()

```
lookup ()
{
  d="$1" ; shift
  f=""
  for k ; do
    v=$(hotplug_get_vpd_value $k)  # Uses $DEVPATH
    [ -n "$v" ] || return
    [ -n "$f" ] && f="${f},"
    f="${f}${v}"
  done
  f=$(clean "$f")
  [ -f "$d/$f" ] && cat "$d/$f"
  exit 0
}
```

vpdnamer - lookup_store()

```
lookup_store ()
{
  ...

  if [ -z "$NAMEDEV_NAMES" ] ; then
    [ -f "$d/$f" ] && cat "$d/$f"
  else
    [ -f "$d/$f" ] || echo "$NAMEDEV_NAMES" > "$d/$f"
  fi
  exit 0
}
```







hdisk namer

/etc/namedev/namers.d/90hdisk:

```
case "$1" in
  sd*|hd*)
    f=/var/state/namedev/hdisk.seq
    if [ -f "$f" ] ; then
      read num <"$f"
      num=$(($num + 1))
    else
      num=0
    fi
    echo $num > "$f"
    echo hdisk$num
esac
```







Does It Work?

- Either 'yes', and talk of a demonstration. . . or
- 'Almost', with a demo of the bits that work.
- I'm hoping to make it work tomorrow!
- **Note:** This slide was written 2 days before the talk. The implementation was a little unstable: the demonstration failed the first time, but worked the second time through. Therefore, I decided to be honest and leave this slide (mostly) as it was.

Will It Work?

- Performance?
- Performance of lsvpd?
- Locking?
- Quality of VPD?







Conclusions

- Nice and general.
- Should be workable.
- Possible performance issues.
- Needed? Is an inconsistent, piecewise approach better?







Questions?

?