AUUG 2003 Conference
Open Standards, Open Source, Open Computing

3 September 2003

Linux hardware inventory: Current reality, future possibilities

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

The plan...

- Introduction.
  - Hardware inventory on Linux®and AIX®.
- Are we there yet?
  - Progress on the Linux *lsvpd* project so far.
- Where to now?
- Conclusions, questions, ...







Current Linux reality

- Linux has hardware inventory systems such as Red Hat's *kudzu* and SuSE's *hwinfo*.







Current Linux reality

- Linux has hardware inventory systems such as Red Hat's *kudzu* and SuSE's *hwinfo*.
- Used mainly for device detection and automated driver selection.







Current Linux reality

- Linux has hardware inventory systems such as Red Hat's *kudzu* and SuSE's *hwinfo*.
- Used mainly for device detection and automated driver selection.
- Information persists between boots.

AIX reality

- Large, mature hardware inventory system.







AIX reality

- Large, mature hardware inventory system.
- Among other things, the Object Data Manager (ODM) contains Vital Product Data (VPD):







AIX reality

- Large, mature hardware inventory system.
- Among other things, the Object Data Manager (ODM) contains Vital Product Data (VPD):
  - Generic, static information about components.
  - Dynamic information about components, including configuration.







AIX reality

- Large, mature hardware inventory system.
- Among other things, the Object Data Manager (ODM) contains Vital Product Data (VPD):
  - Generic, static information about components.
  - Dynamic information about components, including configuration.
- Persistent device naming based on device/slot information (from VPD).







AIX reality

- Large, mature hardware inventory system.
- Among other things, the Object Data Manager (ODM) contains Vital Product Data (VPD):
  - Generic, static information about components.
  - Dynamic information about components, including configuration.
- Persistent device naming based on device/slot information (from VPD).
- `lsvpd` lists VPD in human/machine readable format.
- `lscfg` lists VPD (and other info) in human readable format.







Example of VPD (lsvpd-style)

```
*DS 2 RIO-PCI COPPER
*SN YL1182260007
*PN 53P3820
*CC 2887
*FN  53P3800
*VK RS6K
*YL U0.4-P1.1
```







Example of VPD (explained)

```
*DS 2 RIO-PCI COPPER  # Description
*SN YL1182260007      # Serial Number
*PN 53P3820           # Part Number
*CC 2887
*FN  53P3800          # FRU (Field Replaceable Unit) Number
*VK RS6K
*YL U0.4-P1.1         # Physical Location:
                      # Extender 1, on backplane 1, in drawer 4, in rack 0.
```







Are we there yet?

- General requirement:

Are we there yet?

- General requirement:
  Implement `lsvpd` on Linux.







Are we there yet?

- General requirement:
  Implement `lsvpd` on Linux.
- Various iterations of:
  - more specific requirements
  - 'schedule'
  - choice of implementation language(s)
  - implementation
  - future plans
  - ...







Requirements #1 (May 2001)

- Find `ibm,vpd` properties in the Open Firmware device-tree and pretty print them.
- Time required: a few hours.







Example of PCI VPD

```
82 10 00 32 20 52 49 4f  2d 50 43 49 20 43 4f 50  |...2 RIO-PCI COP|
50 45 52 90 3e 00 53 4e  0c 59 4c 31 31 38 32 32  |PER.>.SN.YL11822|
36 30 30 30 37 50 4e 07  35 33 50 33 38 32 30 43  |60007PN.53P3820C|
43 04 32 38 38 37 46 4e  08 20 35 33 50 33 38 30  |C.2887FN. 53P380|
30 56 4b 04 52 53 36 4b  59 4c 09 55 30 2e 34 2d  |0VK.RS6KYL.U0.4-|
50 31 2e 31 79 ec                                 |P1.1y.|
```

This should take a few hours

- Simple reverse-engineering, parsing and pretty-printing task.







This should take a few hours

- Simple reverse-engineering, parsing and pretty-printing task.
- Single Perl script '`lsvpd`'.







This should take a few hours

- Simple reverse-engineering, parsing and pretty-printing task.
- Single Perl script '`lsvpd`'.
- `ibm,vpd` files in a format that is well defined in the PCI specification.







This should take a few hours

- Simple reverse-engineering, parsing and pretty-printing task.
- Single Perl script '`lsvpd`'.
- `ibm,vpd` files in a format that is well defined in the PCI specification.
- Not so much reverse-engineering.







Requirements #2 (June 2001?)

- 'Things like SCSI devices are missing!'







Requirements #2 (June 2001?)

- 'Things like SCSI devices are missing!'
- Find `ibm,vpd` properties in the Open Firmware device-tree and pretty print them. Also print information about SCSI devices.







Requirements #2 (June 2001?)

- 'Things like SCSI devices are missing!'
- Find `ibm,vpd` properties in the Open Firmware device-tree and pretty print them. Also print information about SCSI devices.
- Time required: a few days.







SCSI standard inquiry output

```
00 00 03 02 9f 00 01 3a  49 42 4d 20 20 20 20 20  |.......:IBM     |
49 43 33 35 4c 30 33 36  55 43 44 32 31 30 2d 30  |IC35L036UCD210-0|
53 35 42 53 56 4d 46 39  39 33 31 38 30 37 4e 34  |S5BSVMF9931807N4|
39 30 38 20 20 20 20 20  0c 00 00 00 00 00 00 00  |908     ........|
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
20 20 30 30 37 35 30 32  31 37 32 00 30 30 30 31  |  007502172.0001|
32 32 30 39 50 33 39 32  33 20 20 20 20 20 48 33  |2209P3923     H3|
32 30 35 31 20 20 20 20  30 37 4e 37 30 37 30 20  |2051    07N7070 |
20 20 20 20 46 38 30 34  37 30 20 20 20 20 32 30  |    F80470    20|
30 32 00 00                                       |02..|
```

Snazzy SCSI solutions

- `update-device-tree`:







Snazzy SCSI solutions

- `update-device-tree`:
  1. Copy `/proc/device-tree` to `/var/lib/device-tree`.







Snazzy SCSI solutions

- `update-device-tree`:
  1. Copy `/proc/device-tree` to `/var/lib/device-tree`.
  2. Use `sg_inq` to do SCSI inquiries.







Snazzy SCSI solutions

- `update-device-tree`:
  1. Copy `/proc/device-tree` to `/var/lib/device-tree`.
  2. Use `sg_inq` to do SCSI inquiries.
  3. Extract 'VPD fields' from output according to templates.







Snazzy SCSI solutions

- `update-device-tree`:
  1. Copy `/proc/device-tree` to `/var/lib/device-tree`.
  2. Use `sg_inq` to do SCSI inquiries.
  3. Extract 'VPD fields' from output according to templates.
  4. Find device-tree node and drop in `linux,vpd` file.







Snazzy SCSI solutions

- `update-device-tree`:
  1. Copy `/proc/device-tree` to `/var/lib/device-tree`.
  2. Use `sg_inq` to do SCSI inquiries.
  3. Extract 'VPD fields' from output according to templates.
  4. Find device-tree node and drop in `linux,vpd` file.
- `lsvpd`:







Snazzy SCSI solutions

- `update-device-tree`:
  1. Copy `/proc/device-tree` to `/var/lib/device-tree`.
  2. Use `sg_inq` to do SCSI inquiries.
  3. Extract 'VPD fields' from output according to templates.
  4. Find device-tree node and drop in `linux,vpd` file.
- `lsvpd`:
  1. Find all files called `ibm,vpd` and render them.







Snazzy SCSI solutions

- `update-device-tree`:
  1. Copy `/proc/device-tree` to `/var/lib/device-tree`.
  2. Use `sg_inq` to do SCSI inquiries.
  3. Extract 'VPD fields' from output according to templates.
  4. Find device-tree node and drop in `linux,vpd` file.
- `lsvpd`:
  1. Find all files called `ibm,vpd` and render them.
  2. Find all files called `linux,vpd` and `cat` them.







Snazzy SCSI solutions

- `update-device-tree`:
  1. Copy `/proc/device-tree` to `/var/lib/device-tree`.
  2. Use `sg_inq` to do SCSI inquiries.
  3. Extract 'VPD fields' from output according to templates.
  4. Find device-tree node and drop in `linux,vpd` file.
- `lsvpd`:
  1. Find all files called `ibm,vpd` and render them.
  2. Find all files called `linux,vpd` and `cat` them.

- This introduced a 'database'.







SCSI standard inquiry output

```
*DS 16 Bit LVD SCSI Disk Drive
*AX /dev/sda
*MF IBM
*TM IC35L036UCD210-0
*YL U0.4-P1-I1/Z2-A8
*FN 09P3923
*RL 53354253
*SN VMF99318
*EC H32051
*PN 07N7070
*Z0 000003029F00013A
*Z1 07N4908
*Z2 0075
*Z3 02172
*Z4 0001
*Z5 22
*Z6 F80470
*Z7 500507620CC4AC8E
```







Perls of wisdom

- Perl's `unpack` function is very useful for parsing binary data, including PCI VPD chunks.

Perls of wisdom

- Perl's `unpack` function is very useful for parsing binary data, including PCI VPD chunks.
- SCSI inquiries reimplemented using Perl's `ioctl` function.







Perls of wisdom

- Perl's `unpack` function is very useful for parsing binary data, including PCI VPD chunks.
- SCSI inquiries reimplemented using Perl's `ioctl` function.
- Finding physical location was problematic:
  - Easy to determine the SCSI host adapter associated with a device.







Perls of wisdom

- Perl's `unpack` function is very useful for parsing binary data, including PCI VPD chunks.
- SCSI inquiries reimplemented using Perl's `ioctl` function.
- Finding physical location was problematic:
  - Easy to determine the SCSI host adapter associated with a device.
  - Need to parse, for example, `/proc/scsi/sym53c8xx/0`.







Perls of wisdom

- Perl's `unpack` function is very useful for parsing binary data, including PCI VPD chunks.
- SCSI inquiries reimplemented using Perl's `ioctl` function.
- Finding physical location was problematic:
  - Easy to determine the SCSI host adapter associated with a device.
  - Need to parse, for example, `/proc/scsi/sym53c8xx/0`.
  - File format is driver-specific.







Perls of wisdom

- Perl's `unpack` function is very useful for parsing binary data, including PCI VPD chunks.
- SCSI inquiries reimplemented using Perl's `ioctl` function.
- Finding physical location was problematic:
  - Easy to determine the SCSI host adapter associated with a device.
  - Need to parse, for example, `/proc/scsi/sym53c8xx/0`.
  - File format is driver-specific.
  - More templates...







Missing bits

- lspci:

```
[...]
241:1.0 SCSI storage controller: LSI Logic / Symbios  53c1010 [...]
[...]
```

- /proc/scsi/sym53c8xx/0:

```
Chip sym53c1010-66, device id 0x21, revision id 0x1
On PCI bus 65, device 1, function 0, IRQ 69
[...]
```

- device-tree:

```
# od -t d /var/lib/lsvpd/device-tree/pci*/bus-range
0000000           0         254           0         254
*
0000460
```







Missing bits

- `lspci`:

```
[...]
241:1.0 SCSI storage controller: LSI Logic / Symbios  53c1010 [...]
[...]
```

- `/proc/scsi/sym53c8xx/0`:

```
Chip sym53c1010-66, device id 0x21, revision id 0x1
On PCI bus 65, device 1, function 0, IRQ 69
[...]
```

- device-tree:

```
# od -t d /var/lib/lsvpd/device-tree/pci*/bus-range
0000000           0         254           0         254
*
0000460
```

- Patched `sym53c8xx_2` driver to print full bus number.
- Patched pSeries™ setup code to drop `linux,phbnum` property in for each PCI host bus.







More Perls of wisdom

- In June 2002, we decided Perl wasn't appropriate for this purpose, since it lives in `/usr`, which might not be available (early enough).







More Perls of wisdom

- In June 2002, we decided Perl wasn't appropriate for this purpose, since it lives in `/usr`, which might not be available (early enough).
- The Perl version retrospectively became a prototype.







More Perls of wisdom

- In June 2002, we decided Perl wasn't appropriate for this purpose, since it lives in `/usr`, which might not be available (early enough).
- The Perl version retrospectively became a prototype.
- However, a scripting language was useful for the main programs.







More Perls of wisdom

- In June 2002, we decided Perl wasn't appropriate for this purpose, since it lives in `/usr`, which might not be available (early enough).
- The Perl version retrospectively became a prototype.
- However, a scripting language was useful for the main programs.
- The templates for describing how to parse SCSI inquiry data were too verbose and difficult to manage.







More Perls of wisdom

- In June 2002, we decided Perl wasn't appropriate for this purpose, since it lives in `/usr`, which might not be available (early enough).
- The Perl version retrospectively became a prototype.
- However, a scripting language was useful for the main programs.
- The templates for describing how to parse SCSI inquiry data were too verbose and difficult to manage.
- Copy of device-tree as a database seemed to work...







More Perls of wisdom

- In June 2002, we decided Perl wasn't appropriate for this purpose, since it lives in `/usr`, which might not be available (early enough).
- The Perl version retrospectively became a prototype.
- However, a scripting language was useful for the main programs.
- The templates for describing how to parse SCSI inquiry data were too verbose and difficult to manage.
- Copy of device-tree as a database seemed to work...
- ...as did the split in functionality between `lsvpd` and `update-device-tree`.







Requirements #3 (June 2002)

- 'Perl isn't around early enough at boot time!'







Requirements #3 (June 2002)

- 'Perl isn't around early enough at boot time!'
- Find `ibm,vpd` properties in the Open Firmware device-tree and pretty print them. Also print information about SCSI devices. Use programming languages that are supported with just a root filesystem.







Requirements #3 (June 2002)

- 'Perl isn't around early enough at boot time!'
- Find `ibm,vpd` properties in the Open Firmware device-tree and pretty print them. Also print information about SCSI devices. Use programming languages that are supported with just a root filesystem.
- Time required: a few weeks.







C & shell (no seashores. . .)

- Scripting languages are good. . .







C & shell (no seashores. . .)

- Scripting languages are good. . .
- . . . but the only scripting language on the root filesystem is the shell.







C & shell (no seashores...)

- Scripting languages are good...
- ...but the only scripting language on the root filesystem is the shell.
- `/bin/bash` can be assumed to be available...







C & shell (no seashores. . .)

- Scripting languages are good. . .
- . . . but the only scripting language on the root filesystem is the shell.
- `/bin/bash` can be assumed to be available. . .
- . . . and has good arithmetic support and other features.







C & shell (no seashores. . .)

- Scripting languages are good. . .
- . . . but the only scripting language on the root filesystem is the shell.
- `/bin/bash` can be assumed to be available. . .
- . . . and has good arithmetic support and other features.
- C chosen for ‘helper utilities’.







C & shell (no seashores. . .)

- Scripting languages are good. . .
- . . . but the only scripting language on the root filesystem is the shell.
- `/bin/bash` can be assumed to be available. . .
- . . . and has good arithmetic support and other features.
- C chosen for ‘helper utilities’.
- *glib-2.0* chosen as a utility library.







Updating `update-device-tree`

- `update-device-tree` is run relatively rarely.
- `lsvpd` is run more often.







Updating `update-device-tree`

- `update-device-tree` is run relatively rarely.
- `lsvpd` is run more often.
- `lsvpd` should be as simple as possible — no rendering — just find `linux,vpd` files and `cat` them.







Updating `update-device-tree`

- `update-device-tree` is run relatively rarely.
- `lsvpd` is run more often.
- `lsvpd` should be as simple as possible — no rendering — just find `linux,vpd` files and `cat` them.
- `update-device-tree` to do all the rendering.







Updating `update-device-tree`

- `update-device-tree` is run relatively rarely.
- `lsvpd` is run more often.
- `lsvpd` should be as simple as possible — no rendering — just find `linux,vpd` files and `cat` them.
- `update-device-tree` to do all the rendering.
- `sed` is my best friend.







Updating `update-device-tree`

- `update-device-tree` is run relatively rarely.
- `lsvpd` is run more often.
- `lsvpd` should be as simple as possible — no rendering — just find `linux,vpd` files and `cat` them.
- `update-device-tree` to do all the rendering.
- `sed` is my best friend.
- Depend on `busybox` for `find` and `sort`.

Rendering ibm,vpd

- pci_vpd_to_txt.[ch]
- ibm_vpd_render.c
- How is ibm,vpd different to PCI VPD?







Rendering SCSI VPD

- Obvious C helper is `sg_inq`.
- Small patch to add `-r` option (raw, binary output) accepted into `sg3_utils` 1.01.







Rendering SCSI VPD

- Obvious C helper is `sg_inq`.
- Small patch to add `-r` option (raw, binary output) accepted into `sg3_utils` 1.01.
- `scsi_vpd_std` (in C) to parse 1st 32 bytes of standard inquiry.







Rendering SCSI VPD

- Obvious C helper is `sg_inq`.
- Small patch to add `-r` option (raw, binary output) accepted into `sg3_utils` 1.01.
- `scsi_vpd_std` (in C) to parse 1st 32 bytes of standard inquiry.
- `scsi_vpd_custom` (in C) to extract custom fields via templates.
- Template format (actually single-line):

```
IBM;disk;*;
inquiry=RL:4,SN:8,Z1:12,_:42,Z2:4,Z3:5,_:1,
    Z4:4,Z5:2,FN:12,EC:10,PN:12,Z6:10,_:4;
0x83=_:8,Z7:8
```







Enter `lscfg`

- 'Human readable' output, plus platform specific information.







Enter `lscfg`

- 'Human readable' output, plus platform specific information.

```
sda              U0.4-P1-I1/Z2-A8  16 Bit LVD SCSI Disk Drive (36400 MB)
      Manufacturer.............IBM
      Machine Type and Model......IC35L036UCD210-0
      Device Specific.(YL)........U0.4-P1-I1/Z2-A8
      FRU Number..................09P3923
      ROS Level and ID............53354253
      Serial Number...............VMF99318
      EC Level....................H32051
      Part Number.................07N7070
      Device Specific.(Z0)........000003029F00013A
      Device Specific.(Z1)........07N4908
      Device Specific.(Z2)........0075
      Device Specific.(Z3)........02172
      Device Specific.(Z4)........0001
      Device Specific.(Z5)........22
      Device Specific.(Z6)........F80470
      Device Specific.(Z7)........500507620CC4AC8E
```

- Initially `lscfg` was a pretty printer.







Cross platforms?

- This could be useful on platforms other than pSeries.







Cross platforms?

- This could be useful on platforms other than pSeries.
- Currently get PCI 2.0/2.1 VPD from device-tree.







Cross platforms?

- This could be useful on platforms other than pSeries.
- Currently get PCI 2.0/2.1 VPD from device-tree.
- Attempted to write `pci_vpd_rom_grab`.







Cross platforms?

- This could be useful on platforms other than pSeries.
- Currently get PCI 2.0/2.1 VPD from device-tree.
- Attempted to write `pci_vpd_rom_grab`.
- Wrote `pci_vpd_cap_grab`.







Testing times (prelude)

- In February 2003, people started testing out the `lsvpd` package...







Requirements #4 (February 2003)

- 'lscfg is very different to the AIX version.'
- 'There's a lot of stuff missing...'
- Find ibm,vpd properties in the Open Firmware device-tree and pretty print them. Also print information about SCSI devices. Use programming languages that are supported with just a root filesystem. Make lscfg work a lot more like the AIX version, implement a whole bunch of options and make more components visible.
- Required time: 6 weeks.

Testing times (summary)

- `lscfg` updated to show platform specific information. Tied more closely to device-tree.
- Synthesised VPD for SCSI and Ethernet adapters from information in the device-tree.







Testing times (summary)

- `lscfg` updated to show platform specific information. Tied more closely to device-tree.
- Synthesised VPD for SCSI and Ethernet adapters from information in the device-tree.
- Distro kernels used *sym53c8xx* driver. Oops...







Testing times (summary)

- `lscfg` updated to show platform specific information. Tied more closely to device-tree.
- Synthesised VPD for SCSI and Ethernet adapters from information in the device-tree.
- Distro kernels used *sym53c8xx* driver. Oops...
- ...added IRQ-matching hack to compensate for broken bus numbers.







Testing times (summary)

- `lscfg` updated to show platform specific information. Tied more closely to device-tree.
- Synthesised VPD for SCSI and Ethernet adapters from information in the device-tree.
- Distro kernels used *sym53c8xx* driver. Oops...
- ...added IRQ-matching hack to compensate for broken bus numbers.
- Released version 0.8.4 as part of SourceForge.net *linux-diag* project in May 2003.







Testing times (summary)

- `lscfg` updated to show platform specific information. Tied more closely to device-tree.
- Synthesised VPD for SCSI and Ethernet adapters from information in the device-tree.
- Distro kernels used *sym53c8xx* driver. Oops...
- ...added IRQ-matching hack to compensate for broken bus numbers.
- Released version 0.8.4 as part of SourceForge.net *linux-diag* project in May 2003.
- Move various bits towards being 'hotplug useful'.







Testing times (summary)

- `lscfg` updated to show platform specific information. Tied more closely to device-tree.
- Synthesised VPD for SCSI and Ethernet adapters from information in the device-tree.
- Distro kernels used *sym53c8xx* driver. Oops...
- ...added IRQ-matching hack to compensate for broken bus numbers.
- Released version 0.8.4 as part of SourceForge.net *linux-diag* project in May 2003.
- Move various bits towards being 'hotplug useful'.
- PCI domain support now in Linux 2.6.
- Under 2.6, use *sysfs* to get PCI information.







Testing times (summary)

- `lscfg` updated to show platform specific information. Tied more closely to device-tree.
- Synthesised VPD for SCSI and Ethernet adapters from information in the device-tree.
- Distro kernels used *sym53c8xx* driver. Oops...
- ...added IRQ-matching hack to compensate for broken bus numbers.
- Released version 0.8.4 as part of SourceForge.net *linux-diag* project in May 2003.
- Move various bits towards being 'hotplug useful'.
- PCI domain support now in Linux 2.6.
- Under 2.6, use *sysfs* to get PCI information.
- 1500 lines of `bash` script and 1500 lines of C source.







Goodbye *glib*!

- The root-filesystem-only requirement meant statically linking to `libglib.a`. Big executables!
- Some code shoe-horned to work with *glib* to make it more maintainable!
- *glib* didn't do everything...







Goodbye *glib*!

- The root-filesystem-only requirement meant statically linking to `libglib.a`. Big executables!
- Some code shoe-horned to work with *glib* to make it more maintainable!
- *glib* didn't do everything...
- Goodbye *glib*!







Goodbye *glib*!

- The root-filesystem-only requirement meant statically linking to `libglib.a`. Big executables!
- Some code shoe-horned to work with *glib* to make it more maintainable!
- *glib* didn't do everything...
- Goodbye *glib*!
- Only had to 'rewrite' a tiny bit of *glib*'s self-expanding string functionality.







Goodbye *glib*!

- The root-filesystem-only requirement meant statically linking to `libglib.a`. Big executables!
- Some code shoe-horned to work with *glib* to make it more maintainable!
- *glib* didn't do everything...
- Goodbye *glib*!
- Only had to 'rewrite' a tiny bit of *glib*'s self-expanding string functionality.
- `asprintf(3)` is a thing of beauty!







Goodbye *glib*!

- The root-filesystem-only requirement meant statically linking to `libglib.a`. Big executables!
- Some code shoe-horned to work with *glib* to make it more maintainable!
- *glib* didn't do everything...
- Goodbye *glib*!
- Only had to 'rewrite' a tiny bit of *glib*'s self-expanding string functionality.
- `asprintf(3)` is a thing of beauty!
- So is `fnmatch(3)`







Cross platforms with *sysfs*

- *sysfs* contains useful information...







Cross platforms with *sysfs*

- *sysfs* contains useful information...
- ...enough for partial implementation of `update-device-tree`...







Cross platforms with *sysfs*

- *sysfs* contains useful information...
- ...enough for partial implementation of `update-device-tree`...
- ...without a device-tree.







Cross platforms with *sysfs*

- *sysfs* contains useful information...
- ...enough for partial implementation of `update-device-tree`...
- ...without a device-tree.
- `lsvpd` even 'runs' on my ThinkPad®.







Self-selecting modules

- Modularised `update-device-tree`, `lsvpd` & `lscfg`.







Self-selecting modules

- Modularised `update-device-tree`, `lsvpd` & `lscfg`.
- Self-selecting modules. For example:

```
/lib/lsvpd/scan.d/30device-tree:
[...]
source_device_tree="/proc/device-tree"
[ -f "${source_device_tree}/system-id" ] || return 0
[...]
```

- Current modules:

```
scan.d/{00minimal,01ethtool,10devfs,
        20sysfs,30device-tree,40ibm,vpd}
lscfg.d/{00minimal,40ibm,vpd}
common.d/00minimal
```

- Subsequent modules redefine `bash` functions from earlier modules.

Future possibilities

- PCI expansion ROM blobs in *sysfs*?





Future possibilities

- PCI expansion ROM blobs in *sysfs*?
- *lsvpd* helping to support persistent device naming.







Future possibilities

- PCI expansion ROM blobs in *sysfs*?
- *lsvpd* helping to support persistent device naming.
- Change management (mostly device/name changes).







Future possibilities

- PCI expansion ROM blobs in *sysfs*?
- *lsvpd* helping to support persistent device naming.
- Change management (mostly device/name changes).
- Large systems ($>$ 128 SCSI disks)?







Future possibilities

- PCI expansion ROM blobs in *sysfs*?
- *lsvpd* helping to support persistent device naming.
- Change management (mostly device/name changes).
- Large systems ($>$ 128 SCSI disks)?
- Larger major/minor numbers?







Future possibilities

- PCI expansion ROM blobs in *sysfs*?
- *lsvpd* helping to support persistent device naming.
- Change management (mostly device/name changes).
- Large systems ($>$ 128 SCSI disks)?
- Larger major/minor numbers?
- *sysfs* support for *sg* driver?







Future possibilities

- PCI expansion ROM blobs in *sysfs*?
- *lsvpd* helping to support persistent device naming.
- Change management (mostly device/name changes).
- Large systems ($>$ 128 SCSI disks)?
- Larger major/minor numbers?
- *sysfs* support for *sg* driver?
- *lsvpd* scalability?







Future possibilities

- PCI expansion ROM blobs in *sysfs*?
- *lsvpd* helping to support persistent device naming.
- Change management (mostly device/name changes).
- Large systems ($>$ 128 SCSI disks)?
- Larger major/minor numbers?
- *sysfs* support for *sg* driver?
- *lsvpd* scalability?
- A standard backend?
  - *sysfs*-based?
  - Common-Information-Model (CIM) Object Manager (or CIMOM)?
  - Database used by *kudzu*?







Conclusions

- Started as a ‘toy’.







Conclusions

- Started as a ‘toy’.
- Now used ‘in anger’.







Conclusions

- Started as a ‘toy’.
- Now used ‘in anger’.
- Lots of work to do. . .







Conclusions

- Started as a ‘toy’.
- Now used ‘in anger’.
- Lots of work to do. . .
- Time required:







Conclusions

- Started as a ‘toy’.
- Now used ‘in anger’.
- Lots of work to do. . .
- Time required: ?







Questions?

?