linux. conf .au
2005
Australian National University, Canberra

22 April 2005

Towards a small, efficient Linux hardware inventory system

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

The Plan. . .

- Introduction.
  - Hardware inventory systems? Small?
  - Output; digging for details.
- VPD-based persistent device naming.
  - udev and `scsi_id`.
  - An lsvpd-based naming system.
- Rewriting lsvpd.
  - Why?
  - Dynamically configurable modules.
  - Function overriding and clever tricks.
  - Infrastructure; Linux® 2.6 only?
- Scripting system tools?
- Conclusions, questions, . . .

Hardware Inventory Systems? Small?

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

Vital Product Data

```
*DS PCI-X Dual Channel Ultra320 SCSI RAID Adapter
*AX scsi3
*PN 97P3960
*FN 97P3960
*SN YL10C3306827
*MN 000C
*EC 0
*RM 0309002d
*Z0 5703
*Z1 1
*YL U7879.001.11C543F-P1-C5-T1
```

Vital Product Data — Explained

```
*DS PCI-X Dual Channel Ultra320 SCSI RAID Adapter
*AX scsi3                          # OS/AIX Name
*PN 97P3960                        # Part Number
*FN 97P3960                        # FRU Number
*SN YL10C3306827                   # Serial Number
*MN 000C                           # Manufacturer ID
*EC 0                              # Engineering Level
*RM 0309002d                       # ROM Level
*Z0 5703
*Z1 1
*YL U7879.001.11C543F-P1-C5-T1 # Physical Location
```

lscfg Output

```
scsi3            U7879.001.11C543F-P1-C5-T1
                                      PCI-X Dual Channel Ultra320 SCSI RAID
                                      Adapter

      Part Number.................97P3960
      FRU Number..................97P3960
      Serial Number...............YL10C3306827
      Manufacture ID..............000C
      EC Level....................0
      Alterable ROM Level.........0309002d
      Device Specific.(Z0)........5703
      Device Specific.(Z1)........1
      Device Specific.(YL)........U7879.001.11C543F-P1-C5-T1
```

Vital Product Data — Sources

```
*DS PCI-X Dual Channel Ultra320 SCSI RAID Adapter
*AX scsi3                        # Linux/sysfs
*PN 97P3960                      # PCI 2.0/2.1 VPD
*FN 97P3960                      # PCI 2.0/2.1 VPD
*SN YL10C3306827                 # PCI 2.0/2.1 VPD
*MN 000C                         # PCI 2.0/2.1 VPD
*EC 0                            # PCI 2.0/2.1 VPD
*RM 0309002d                     # Linux/sysfs
*Z0 5703
*Z1 1
*YL U7879.001.11C543F-P1-C5-T1 # OF device-tree

# DS from OF device-tree (PCI 2.0, 2.1 VPD)
```

Persistent Device Naming (SCSI-only)

- udev
- `scsi_id`
- Run script to create rules.

VPD-based Persistent Device Naming

1. `lsvpd-hotplug` retrieves VPD.
2. udev calls `lsvpd-namedev`.
   (a) Do we have a name for various combinations of VPD fields?
       If so, return it.
   (b) If not, generate one from a sequence.
       Associate name with combination of VPD fields.
       Return it.
3. `lsvpd-hotplug-name` adds the name to VPD.

Rewriting **lsvpd**

- Currently `bash` + C helpers.
- Inefficient.
- Run-time dependencies.

Dynamically Configurable Modules (1)

- Directory of modules, sourced in order, `run-parts`-style.
- Modules check their own ‘load condition’.
- Easy in `bash`:

```
[ -n "$sysfs_dir" ] || return 0

list_devices_functions="sysfs_list_devices"

sysfs_list_devices ()
{
    ...
}
```

Dynamically Configurable Modules (2)

- In C, use *ELF sections*.
- ELF sections combined in link (`ld`) order.
- Check condition in a function, and perform 'magic' on function:

```
static void
init(void)
{
        if (NULL != lsvpd_sysfs_dir) {
                device_listing_functions_clear();
                ...
        }
}

INIT(init);
```

Dynamically Configurable Modules (3)

- Line noise:

```
#define INIT(fn) static initcall_t __initcall_##fn \
__attribute__((__unused__)) \
__attribute__((__section__("init_call"))) = &fn

static inline void call_inits (void)
{
        extern initcall_t __start_init_call[], __stop_init_call[];
        initcall_t *p;
        for (p = __start_init_call; p < __stop_init_call; p++)
                (*p)();
}
```

Dynamically Configurable Modules (3)

- Line noise:

```
#define INIT(fn) static initcall_t __initcall_##fn \
__attribute__((__unused__)) \
__attribute__((__section__("init_call"))) = &fn

static inline void call_inits (void)
{
        extern initcall_t __start_init_call[], __stop_init_call[];
        initcall_t *p;
        for (p = __start_init_call; p < __stop_init_call; p++)
                (*p)();
}
```

- Alternatives:
  - `.init_array` section.
  - `constructor` attribute.

Function overriding

- In `bash` redefine function in later modules.
- In C use function pointers.

Function overriding

- In `bash` redefine function in later modules.
- In C use function pointers.
- Function multiplexing in bash:
  - A SCSI device is added.
  - `device_add scsi 0:0:8:0`
    1. Try `device_add_scsi 0:0:8:0`
    2. Try `device_add_DEFAULT scsi 0:0:8:0`
    3. Do nothing!

Function multiplexing in C

- Array of functions per multiplexed function?

Function multiplexing in C

- Array of functions per multiplexed function? No!

Function multiplexing in C

- Array of functions per multiplexed function? No!
- List of structures per function type.
  - Structure contains:
    - ∗ Function pointers.
    - ∗ Function type field (string).

Function multiplexing in C

- Array of functions per multiplexed function? No!
- List of structures per function type.
  - Structure contains:
    - Function pointers.
    - Function type field (string).
  - Can unset functions.
  - Need to hand-code top-level function, or use unwieldy macros.

Infrastructure

- Language infrastructure:
  - libc?
  - glib?
  - Kernel list 'library'?
  - String handling? `asprintf(3)`!

Infrastructure

- Language infrastructure:
  - libc?
  - glib?
  - Kernel list 'library'?
  - String handling? `asprintf(3)`!
- Project infrastructure:
  - In `bash` version of lsvpd, can add new features in minutes.
  - Domain-specific infrastructure.

Infrastructure

- Language infrastructure:
  - libc?
  - glib?
  - Kernel list 'library'?
  - String handling? `asprintf(3)`!
- Project infrastructure:
  - In `bash` version of lsvpd, can add new features in minutes.
  - Domain-specific infrastructure.
- Linux 2.6 only?

Scripting System Tools

- Sophisticated scripting languages have non-trivial run-time dependencies.
- People that try to build on top of your tool inherit those dependencies.

Conclusions

- Several iterations of Linux lsvpd.
- Linux lsvpd is getting better.
- Support persistent device naming on large systems?
- Scripting languages can't be used to solve all problems.

Questions?

?