QEMU's new device model qdev

Markus Armbruster
armbru@redhat.com

KVM Forum 2010

Before qdev: No Common Device Model

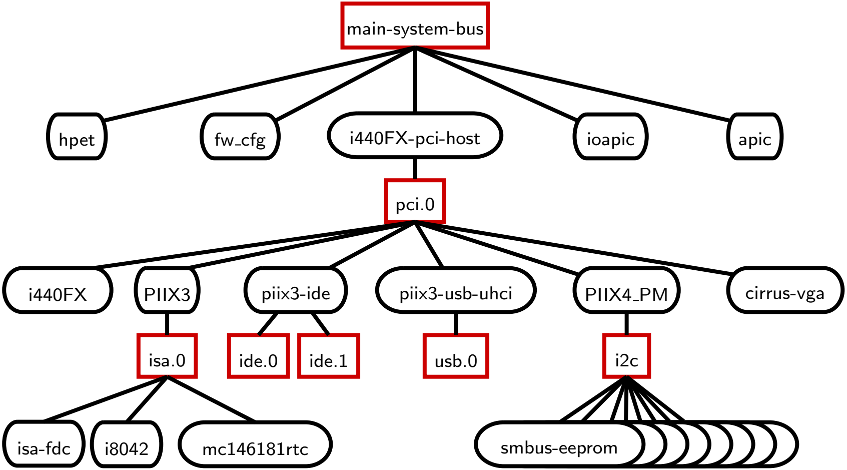
You are in a maze of twisty little devices,
all different.

```
-drive if=TYPE,index=IDX,bus=BUS,unit=UNIT,HOST-OPTS...
-usbdevice disk:format=FMT:FILENAME
-serial CHARDEV
-parallel CHARDEV
-usbdevice serial:vendorid=VID,productid=PRID,CHARDEV
-usbdevice NAME
-virtioconsole CHARDEV
-net nic,vlan=VLAN,macaddr=MACADDR,model=MODEL,name=ID,addr=STR,vectors=V
-usbdevice net:vlan=VLAN,macaddr=MACADDR,name=ID,addr=STR,vectors=V
-vga VGA
-soundhw C1,...
-watchdog NAME
-pcidevice host=ADDR,dma=none,id=ID
-usbdevice host:auto:BUS.ADDR:VID:PRID
```

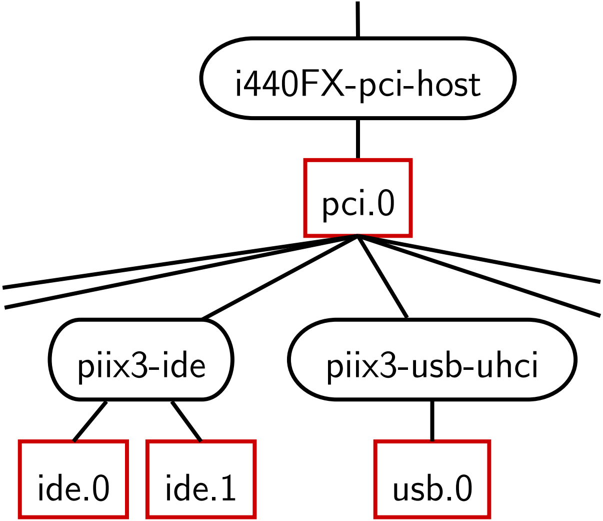
With qdev: Device Model Abstraction

- Tree of devices connected by buses
- Devices have properties
- Devices implement a common API
- Generic device configuration & control
- Turn code into data
- qdev is conceptually simple, devil's in the details

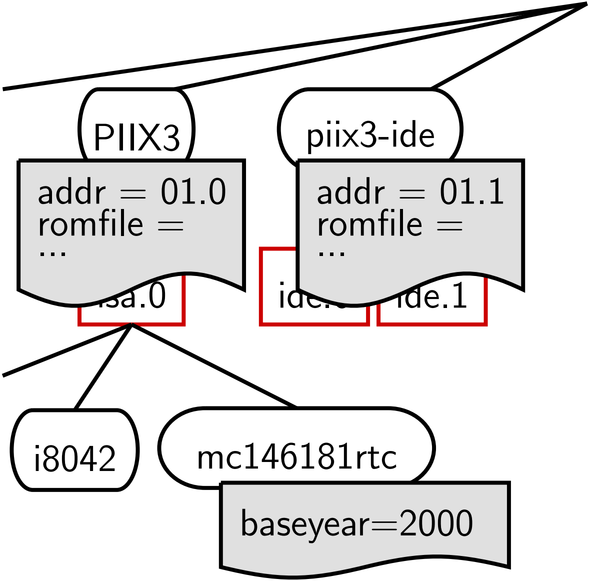
Example Device Tree



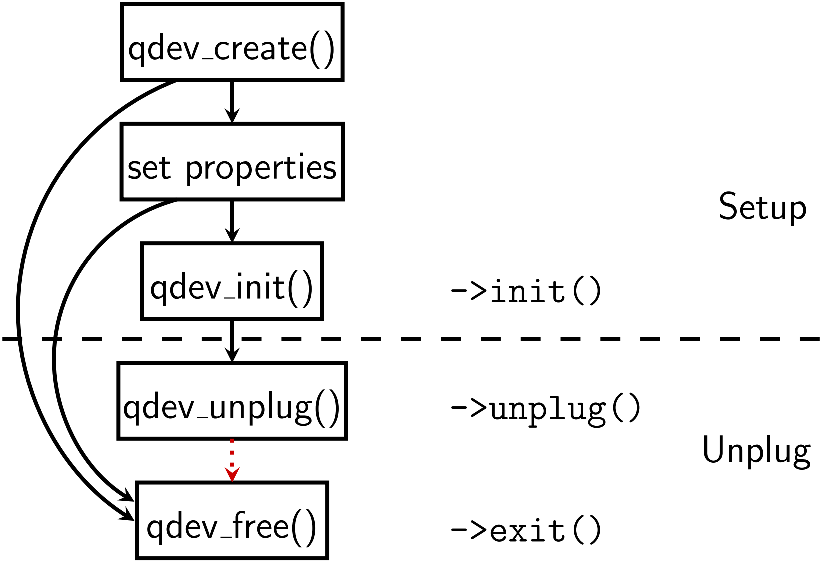
Example Device Tree: Zoom in



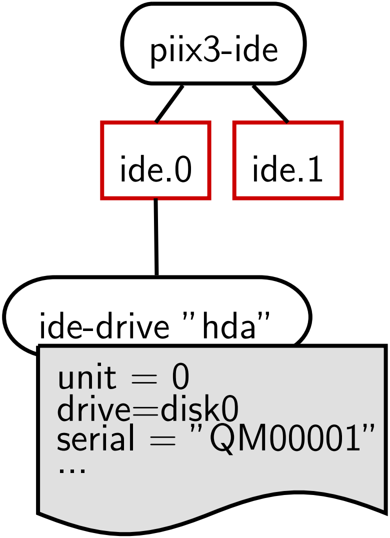
Example Device Tree: Properties



Device Live Cycle



Device Setup



- Command line:

```
-device ide-drive,...
```

- Monitor (hot plug):

```
device_add ide-drive,...
```

- Configuration file:

```
[device "hda"]
  driver = "ide-drive"
  bus = "ide.0"
  unit = "0"
  drive = "disk0"
```

Device Unplug

- Monitor (hot unplug):
  `device_del DEV-ID`
- Bus needs to support hot unplug
- For PCI, only starts ACPI dance
  (we should report completion somehow)

Naming Devices and Buses

Device:

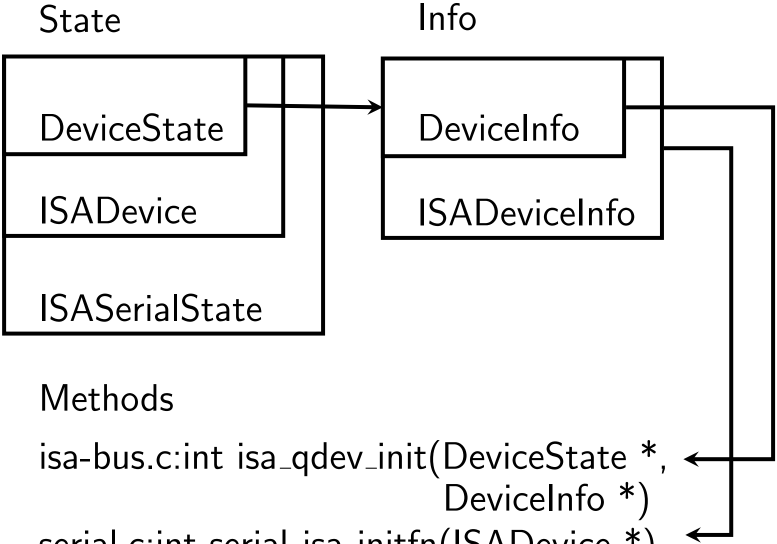
- By unique device ID (chosen by user)

Bus:

- By bus ID (possibly ambiguous)
- By path (messed up, do not use)

Wanted: sane device tree paths

Anatomy of a Simple Device: isa-serial



State

Info

DeviceState

ISADevice

ISASerialState

DeviceInfo

ISADeviceInfo

Methods

isa-bus.c:int isa_qdev_init(DeviceState *,
DeviceInfo *)

serial.c:int serial_isa_initfn(ISADevice *)

Device Anatomy: State

```
typedef struct ISASerialState {
    ISADevice dev;
    uint32_t index;
    uint32_t iobase;
    uint32_t isairq;
    SerialState state;
} ISASerialState;
```

ISASerialState extends bus's ISADevice
extends generic DeviceState

Device Anatomy: Info

```
static ISADeviceInfo serial_isa_info = {
    .qdev.name  = "isa-serial",
    .qdev.size  = sizeof(ISASerialState),
    .qdev.vmsd  = &vmstate_isa_serial,
    .init       = serial_isa_initfn,
    .qdev.props = ...
};
```

`ISADeviceInfo` extends generic `DeviceInfo`

`serial_isa_info` describes `ISASerialState`
and provides methods

Device Anatomy: Properties

Info member qdev.props is:

```
(Property[]) {
    DEFINE_PROP_UINT32("index",
            ISASerialState, index,  -1),
    DEFINE_PROP_HEX32("iobase",
            ISASerialState, iobase, -1),
    ...
}
```

Describe configurable members of state

Using poor man's reflection

redhat

Device Anatomy: vmstate

Info member qdev.vmsd points to:

```
static VMStateDescription vmstate_isa_serial = {
    ...
    .fields = (VMStateField []) {
        VMSTATE_STRUCT(state, ISASerialState, ...)
        ...
    }
};
```

Describes persistent members of state

Poor man's reflection again (no code shared)

Device Anatomy: Methods

- Available methods depend on bus
- ISA: just `ISADeviceInfo` method `init()`

```
static int
serial_isa_initfn(ISADevice *dev)
{
    Check properties
    Start up device model
    return 0;
}
```

How to qdevify a Simple Device

- State: make it extend bus's device state
- Info: name, size, properties, vmstate, . . .
- Refactor code for device info methods
- Turn legacy interfaces into sugar
- Study existing conversions (git is your friend)

Future Work

- Self-documentation
- Clean up a few messes
- Device classification
- Qdevify or shoot the stragglers
- Turn more code into data

Questions?