Finding your way through the QEMU parameter jungle

2018-02-04

Thomas Huth <thuth@redhat.com>
Legal

● Disclaimer: Opinions are my own and not necessarily the views of my employer

● “Jungle Leaves” background license: CC BY 3.0 US:
https://creativecommons.org/licenses/by/3.0/us/
Image has been modified from the original at:
https://www.freekvectors.com/jungle-leaves-vector-background
Introduction
Why a guide through the QEMU parameter jungle?
Why a guide through the QEMU parameter jungle?

- QEMU is a big project, supports lots of emulated devices, and lots of host backends
- 15 years of development → a lot of legacy
- `$ qemu-system-i386 -h | wc -l` 454
- People regularly ask about CLI problems on mailing lists or in the IRC channels

→ Use libvirt, virt-manager, etc. if you just want an easier way to run a VM
General Know-How

- QEMU does not distinguish single-dash options from double-dash options:
  \[-h = --h = -\text{help} = --\text{help}\]

- QEMU starts with a set of default devices, e.g. a NIC and a VGA card. If you don't want this:
  \[--\text{nodefaults}\]
  or suppress certain default devices:
  \[--\text{vga none}\]
  \[--\text{net none}\]
Getting help about the options

- Parameter overview:
  - `--h` or `--help` (of course)

- Many parameters provide info with “help”:
  - `--accel help`

- Especially, use this to list available devices:
  - `--device help`

- To list parameters of a device:
  - `--device e1000,help`

- To list parameters of a machine:
  - `--machine q35,help`
e1000 example

- $ qemu-system-x86_64 --device e1000,help
  [...]  
  e1000.addr=int32 (PCI slot and function...)  
e1000.x-pcie-extcap-init=bool (on/off)  
e1000.extra_mac_registers=bool (on/off)  
e1000.mac=str (Ethernet 6-byte MAC Address...)  
e1000.netdev=str (ID of a netdev backend)

- $ qemu-system-x86_64 --device \  
  e1000,mac=52:54:00:12:34:56,addr=06.0
General Know How: Guest and Host

There are always two parts of an emulated device:

- Emulated guest hardware, e.g.: --device e1000
- The backend in the host, e.g.: --netdev tap

Make sure to use right set of parameters for configuration!
“Classes” of QEMU parameters

- **Convenience**: Easy to use, but often limited scope. For example:
  ```bash
  --cdrom filename
  ```

- **Architected**: Full control over internals, but often cumbersome to use directly. E.g.:
  ```bash
  --device ide-cd,... --blockdev raw,...
  ```

- **Legacy**: Mostly still there to stay compatible with older versions of QEMU
  For example:
  ```bash
  --drive if=ide,media=cdrom,...
  ```
Character devices
Character devices

• List available backends with:
  --chardev help
• E.g.: file, socket, stdio, pipe, tty, …
• To redirect a serial port to a file:

$ qemu-system-x86_64 \n   --nodefaults --nographic \n   --chardev file,id=c1,path=io.txt \n   --device isa-serial,chardev=c1
Chardevs – legacy options

• Legacy / convenience options, for example: --serial and --parallel

• Can be useful for boards with serial output:
  $ qemu-system-ppc -M ppce500
     --nographic --nodefaults
     --serial mon:stdio

• mon:stdio gives you the QEMU monitor and the guest serial output on stdio (toggle with CTRL-a c)
Network devices
Modern network devices

- Use `--netdev type, id=id, ...` to configure the backend.

- Type can be for example:
  - `user`: “emulated” net stack (no privileges required)
  - `tap`: “real” network connection via bridge
  - `socket`: tunnel via a socket to other QEMU

- Example:
  ```
  $ qemu-system-x86_64 \
  --device virtio-net,netdev=n1 \
  --netdev user,id=n1,dhcpstart=10.0.0.50
  ```
Legacy network devices

• Both, device and backend via -net:
  -net nic,model=e1000,vlan=0
  -net user,vlan=0

• vlan is not IEEE 802.1Q – it's a network hub number!

• Better use --netdev if possible! However:
  -net is still the only way to configure some on-board NICs (on embedded boards)
--netdev vs. -net

With --netdev you get simple 1:1 connections:

```bash
--netdev user,id=n1 \ 
--device e1000,netdev=n1 \ 
--netdev tap,id=n2 \ 
--device virtio-net,netdev=n2
```
--netdev vs. -net

With -net you get a hub inbetween:
-\texttt{-net nic, model=e1000 \}
-\texttt{-net user \}
-\texttt{-net nic, model=virtio \}
-\texttt{-net tap}

\begin{center}
\begin{tikzpicture}
\node (e0000) at (0,0) {e0000};
\node (user) at (2,0) {user};
\node (virtio-net) at (4,0) {virtio-net};
\node (tap) at (6,0) {tap};
\node (hub) at (0,-2) {Hub #0 ("vlan=0")};
\draw [->] (e0000) -- (user);
\draw [->] (user) -- (virtio-net);
\draw [->] (virtio-net) -- (tap);
\draw [->] (hub) -- (e0000);
\draw [->] (hub) -- (user);
\draw [->] (hub) -- (virtio-net);
\draw [->] (hub) -- (tap);
\end{tikzpicture}
\end{center}
Block devices
Block devices – the modern way

- Use **--blockdev** to configure a block backend, for example:

```
$ qemu-system-x86_64 -m 1G \
   --blockdev file,node-name=f1,filename=img.qcow2 \
   --blockdev qcow2,node-name=q1,file=f1 \
   --device ide-hd,drive=q1
```
Block devices – the modern way

- Use `--blockdev` to configure a block backend, for example:

```bash
$ qemu-system-x86_64 -m 1G \
   --blockdev file,node-name=f1,filename=img.qcow2 \
   --blockdev qcow2,node-name=q1,file=f1 \
   --device ide-hd,drive=q1
```

- Or shorter:

```bash
$ qemu-system-x86_64 -m 1G \
   --blockdev qcow2,node-name=q2,file.driver=file,\  
   file.filename=img.qcow2 \
   --device ide-hd,drive=q2
```

- ⇒ fine-grained control over the block stack
What is possible today

(slid taken from “More Block Device Configuration” presentation at KVM Forum 2014 by Max Reitz and Kevin Wolf)
The legacy drive option

- Use `--drive` to create the guest device and backend together, e.g.:
  
  ```
  $ qemu-system-x86_64 \
      --drive if=ide,format=raw,file=disk.img
  ```

- “`if=none`” to only configure the backend (used before `--blockdev` was introduced)

- Legacy parts might be removed – don't use `--drive` anymore in new scripts / code!
The convenience options

- `--hda ... --hdd` for hard disks
- `--fda` and `--fdb` for floppy devices
- `--cdrom` for CD-ROM devices
- `--pflash` for parallel flash block devices
- Etc.
Things to remember

- QEMU devices consist of two parts: Emulated hardware and host backends

- Use `--chardev`, `--netdev` and `--blockdev` together with `--device` for full control of the QEMU internals

- Avoid legacy options like `–net` and `–drive` in scripts and places that should be future-proof
Thank you!

Questions?
Resources

- The QEMU documentation: https://qemu.weilnetz.de/doc/qemu-doc.html

- The documentation section in the Wiki: https://wiki.qemu.org/Documentation

- Examples for certain features: https://wiki.qemu.org/Features

- Block Device configuration presentations: