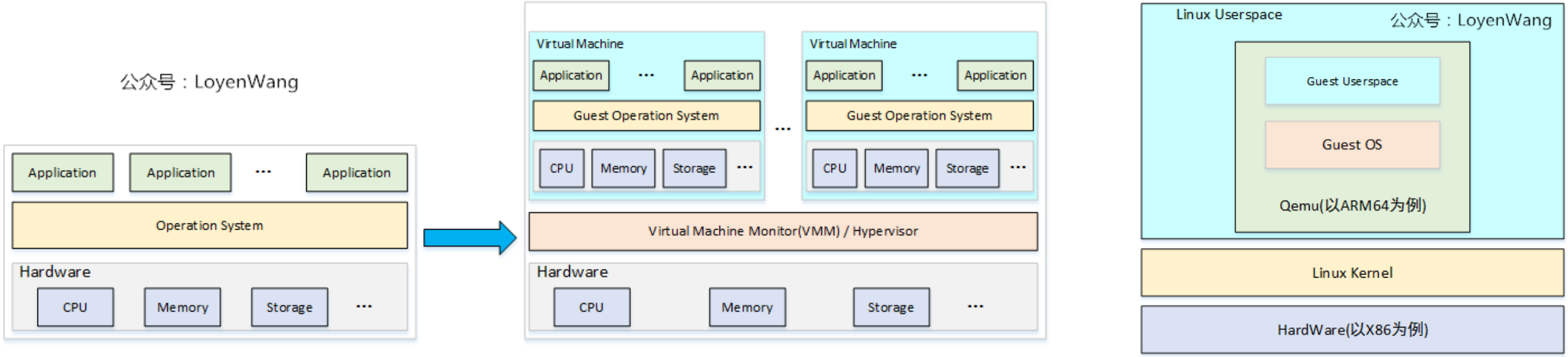
qemu

Du Jiajun

- qemu的工作方式
- 启动一个kernel
- 添加块设备的方式
- 网络配置的方式
- 共享宿主机的目录

qemu的工作方式

qemu的工作方式：



qemu有两种工作的模式：全系统模拟模式（full-system emulation）、用户模式（Usermode emulation）。

用户模式的工作方式：只负责翻译elf格式的可执行文件，通过翻译系统调用的方式使用host内核和资源，可以看作是一个执行器。命令是qemu-{guest_arch}，如qemu-x86_64.

qemu是一种软件虚拟化，即通过软件模式来实现VMM（Virtual Machine Monitor）层。在没有硬件虚拟化的支持下，QEMU本质上完成的工作是二进制的翻译，将guest OS的代码指令翻译成host架构支持的代码指令，模拟出一个完整的系统，包含处理器以及各种外围设备。命令是qemu-system-{guest-arch}，如qemu-system-aarch64.

qemu的工作方式：

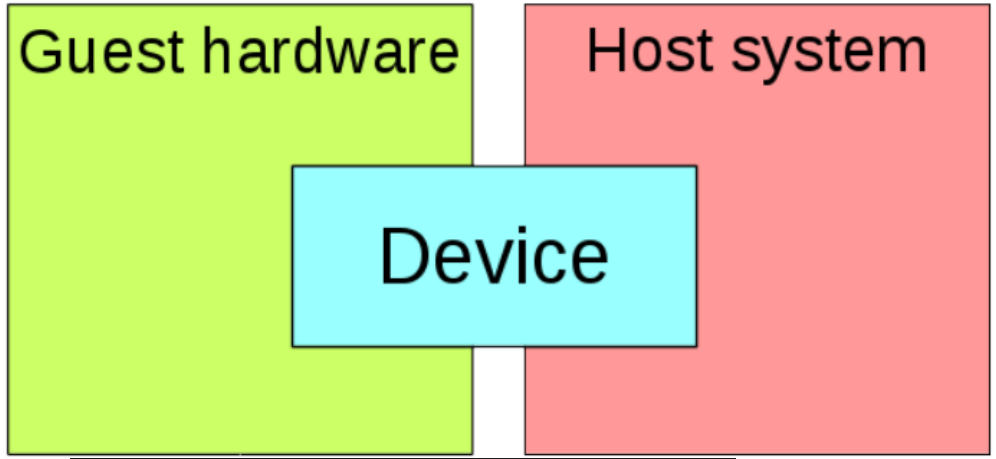
在qemu中定义设备的基本的模式是：backend+device。

backend是指qemu如何处理这个设备如何和host主机进行交互，也叫后端，描述了在仿真中如何使用主机的资源；后端的选择通常对访客来说是透明的。

device是定义在虚拟机内部可以看到的设备参数，也叫前端，向访客呈现设备的方式；所有设备的前端都可以使用 --device参数指定。

使用--device help命令就能列出在对应的架构和机器类型上支持的所有设备类型。

qemu中的参数不区分--和-。




```
Storage devices:
name "am53c974", bus PCI, desc "AMD Am53c974 PCscsi-PCI SCSI adapter"
name "cxl-type3", bus PCI, desc "CXL PMEM Device (Type 3)"
name "dc390", bus PCI, desc "Tekram DC-390 SCSI adapter"
name "ich9-ahci", bus PCI, alias "ahci"
name "ide-cd", bus IDE, desc "virtual IDE CD-ROM"
name "ide-hd", bus IDE, desc "virtual IDE disk"
name "lsi53c810", bus PCI
name "lsi53c895a", bus PCI, alias "lsi"
name "megasas", bus PCI, desc "LSI MegaRAID SAS 1078"
name "megasas-gen2", bus PCI, desc "LSI MegaRAID SAS 2108"
name "mptsas1068", bus PCI, desc "LSI SAS 1068"
name "nand"
name "nvdimm", desc "DIMM memory module"
name "nvme", bus PCI, desc "Non-Volatile Memory Express"
name "nvme-ns", bus nvme-bus, desc "Virtual NVMe namespace"
name "nvme-subsys", desc "Virtual NVMe subsystem"
name "pvscsi", bus PCI
name "scsi-block", bus SCSI, desc "SCSI block device passthrough"
name "scsi-cd", bus SCSI, desc "virtual SCSI CD-ROM"
name "scsi-generic", bus SCSI, desc "pass through generic scsi device (/dev/sg*)"
name "scsi-hd", bus SCSI, desc "virtual SCSI disk"
name "sd-card", bus sd-bus
name "sdhci-pci", bus PCI
name "usb-bot", bus usb-bus
name "usb-mtp", bus usb-bus, desc "USB Media Transfer Protocol device"
name "usb-storage", bus usb-bus
name "usb-uas", bus usb-bus
name "vhost-scsi", bus virtio-bus
name "vhost-scsi-pci", bus PCI
name "vhost-scsi-pci-non-transitional", bus PCI
name "vhost-scsi-pci-transitional", bus PCI
name "vhost-user-blk", bus virtio-bus
name "vhost-user-blk-pci", bus PCI
name "vhost-user-blk-pci-non-transitional", bus PCI
name "vhost-user-blk-pci-transitional", bus PCI
name "vhost-user-fs-device", bus virtio-bus
name "vhost-user-fs-pci", bus PCI
name "vhost-user-scsi", bus virtio-bus
name "vhost-user-scsi-pci", bus PCI
name "vhost-user-scsi-pci-non-transitional", bus PCI
name "vhost-user-scsi-pci-transitional", bus PCI
name "virtio-blk-device", bus virtio-bus
name "virtio-blk-pci", bus PCI, alias "virtio-blk"
name "virtio-blk-pci-non-transitional", bus PCI
name "virtio-blk-pci-transitional", bus PCI
name "virtio-scsi-device", bus virtio-bus
name "virtio-scsi-pci", bus PCI, alias "virtio-scsi"
name "virtio-scsi-pci-non-transitional", bus PCI
name "virtio-scsi-pci-transitional", bus PCI
```

启动一个kernel

（-boot参数是用来指定启动的顺序，如d表示从CD-ROM开始启动系统，一般可以用来将系统安装到一个虚拟磁盘文件上。）

启动一个系统：

一、最常用
1.准备一个kernel的镜像，通过-kernel参数指定文件，比如Image、zImage、bzImage，可以通过-append参数指定内核的参数，比如root=(root file system)
2.如何指定BIOS：
　　（1）qemu默认使用SeaBios作为BIOS
　　（2）使用-bios参数指定BIOS
3.如何加载rootfs：
　　（1）直接通过-append参数指定root的设备名称
　　（2）通过使用-initrd参数指定initial ram disk进行加载
4.可以选择传递dtb设备树的镜像，在boot的时候加载硬件信息。

比如：

```
qemu-system-aarch64
    -m 1024 -M raspi3b -smp 4 -nographic
    -kernel kernel8.img -append "console=ttyAMA0 root=/dev/mmcblk0p2 rw rootwait rootfstype=ext4"
    -dtb bcm2710-rpi-3-b-plus.dtb
    -sd 2022-01-28-raspios-bullseye-arm64.img
    -device usb-net,netdev=net0 -netdev user,id=net0,hostfwd=tcp::5555-:22
```

二、
-device loader,file=<file>[,addr=<addr>][,cpu-num=<cpu-num>][,force-raw=<raw>]
可以直接将elf格式的kernel加载到addr地址上。
应该rCore的实验里面是这样加载的，还没看。

三、
如果在磁盘镜像上安装了系统，那么就可以直接从磁盘开始启动：

安装操作系统 [编辑 | 编辑源代码]

这是你第一次需要去启动模拟器的步骤，为了在磁盘镜像上安装操作系统，你必须同时将磁盘镜像与安装介质装载到虚拟机上，从安装介质中启动操作系统。

以i386的客户机为例，为了从CD-ROM内的把可用于启动的ISO文件安装到磁盘镜像上，你需要：

```
$ qemu-system-x86_64 -cdrom iso_image -boot order=d -drive file=disk_image,format=raw
```

参阅 **qemu(1)** 获得更多关于不同类型安装介质的信息 (例如floppy，磁盘镜像和物理驱动盘)，参阅 **#运行虚拟化的系统** 了解更多有用的选项。

在安装完操作系统后，就可以直接从QEMU镜像内启动了。（参阅 **#运行虚拟化的系统** ）

注意： 默认情况下仅分配给虚拟机128MiB的内存， 分配的内存大小可以通过 `-m` 调整， 比如 `-m 512M` 或 `-m 2G`。

提示：

- 相较于指定 `-boot order=x` ，一部分用户感觉使用 `-boot menu=on` 启用boot菜单的体验更舒服些，至少在配置和实验时是这样的。
- 当使用无界面（headless）模式时， 将会默认在本地5900端口启动一个VNC服务器， 可以用 **TigerVNC** 连接到客户机的系统上:
  `vncviewer :5900`
- 若你在安装过程中需要替换软盘或CD，可以使用QEMU机器监视器（在虚拟机窗口中按 `Ctrl + Alt + 2` ）来删除存储设备并将其连接到虚拟机。使用 `info block` 查看块设备，然后使用 `change` 命令换出设备。按下 `Ctrl + Alt + 1` 返回虚拟机。

运行虚拟化的系统 [编辑 | 编辑源代码]

`qemu-system-*` 程序 (例如 `qemu-system-i386` 或 `qemu-system-x86_64`，取决于客户机架构)用来运行虚拟化的客户机. 用法是:

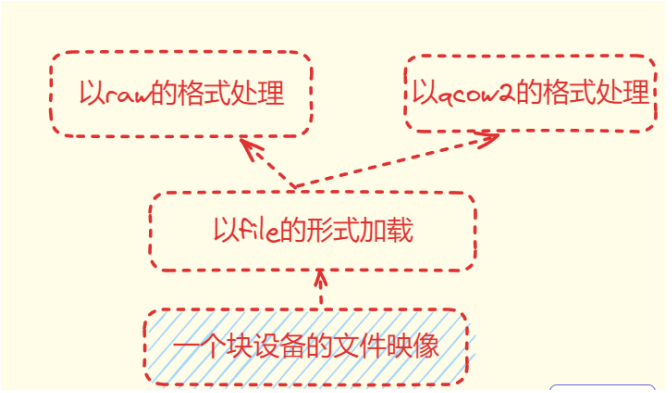
```
$ qemu-system-i386 options disk_image
```

添加块设备的方式

1.一个官方保证稳定、推荐的定义方式：-blockdev和-device结合

-blockdev：定义后端，生成一个块设备的节点，指定qemu如何处理host主机上的资源；
-device：定义前端，指定虚拟机中可以看到的设备的信息；

-blockdev参数中还有一个子参数-driver，用来定义qemu如何加载块设备，需要分级加载（细粒度的管理）：
（1）-driver=file：首先需要通过以file的形式加载进来，然后在以其他的方式进行处理。
（2）-driver=raw：以raw的格式去加载块设备。
（3）-driver=qcow2：以qcow2的格式去加载块设备。
（4）...



比如rros常用的2022-01-28-raspios-bullseye-arm64-rros.img镜像，就可以使用以下方法进行加载：

```
-blockdev driver=file,node-name=my_file,filename=../2022-01-28-raspios-bullseye-arm64-rros.img
    -blockdev driver=raw,node-name=ddd,file=my_file
    -device virtio-blk-device,drive=ddd
```

这里分开加载了file和raw，也可以使用inline的写法：

```
-blockdev driver=raw,node-name=my_file,file.driver=file,file.filename=../2022-01-28-raspios-bullseye-arm64-rros.img
    -device virtio-blk-device,drive=my_file
```

2.qemu提供了一个-blockdev和-device写法的shortcut，就是-drive：

-drive：简化了使用-blockdev分级加载的方式，使用format参数代替。

子参数-if：定义连接的接口，主要有ide、scsi、sd、floppy。定义为none就是该设备将不会被模拟器识别和驱动，而是作为原始设备或裸设备进行操作，这时-drive就只能定义设备后端，仍然需要使用device定义设备前端。

子参数-format定义加载的方式。

子参数-media用来定义该镜像用来模拟什么，disk表示模拟一个硬盘，cdrom表示模拟一个光盘，通常默认是disk。

比如上面的命令也可以写成：

-drive if=none,file=../2022-01-28-raspios-bullseye-arm64-rros.img,id=hd0,format=raw,media=disk
-device virtio-blk-pci,drive=hd0

另外这样就同时定义了前端和后端：
qemu-system-x86_64 -drive if=ide,format=raw,file=disk.img

以下是一些为了方便而衍生出来的参数

3.（老参数，不建议使用）-hda、-hdb、-hdc、-hdd：表示将文件加载进来，将该文件加载成硬盘的镜像文件。abcd表示该硬盘连接的接口顺序。可以认为是当-media=disk的时候的一个shortcut。

-hda ../2022-01-28-raspios-bullseye-arm64-rfl.img

会自动检测镜像文件的format，但是会给出一个没有指定format=raw的warning

4.-fda、-fdb：定义floppy接口的块设备。可以认为是-drive if=floppy的shortcut。

5.-cdrom：定义cdrom设备。可以认为是-media=cdrom时的shortcut。

6.-pflash：定义pflash接口的块设备。可以认为是-drive if=pflash的shortcut。

7.-sd：添加一个sd卡的文件镜像

...

网络配置的方式

网络配置的方式

1.user模式：

用户协议栈方式，原理是在qemu的进程中实现一个协议栈，这个协议栈被视为host和qemu之间的NAT（network address translation）服务器，将qemu所模拟的系统的网络请求转发到host的网卡上面，从而实现网络通信。

使用简单，无需对host主机进行额外的配置。但是没有客户机无法支持全部的ip协议，支持TCP、UDP协议，但是不支持ICMP协议。

配置的方式：

（1）-netdev和-device的组合：

参数-netdev用于指定虚拟网络设备的后端，即定义host和虚拟机之间的通信方式，有user、tap、bridge、socket等。

参数-device用于设置虚拟机内部可见的设备参数，将该设备连接到id指定的netdev上；定义多个-device就可以通过ifconfig看到多个网卡。

```
-device {net device model},netdev=net0 \
-netdev user,id=net0,hostfwd=tcp:127.0.0.1:2222-:22,hostfwd=::2223-:23
```

（可以使用hostfwd定义多个端口映射，hostfwd=[tcp|udp]:[hostaddr]:hostport-[guestaddr]:guestport）

（使用qemu-system-aarch64 -machine type=virt -device help查看）

（2）-nic是一个简化版的定义方式：

```
-nic user,model={net device model},hostfwd=::2222-:22
```

网络配置的方式

（3）-netdev和-net的组合：

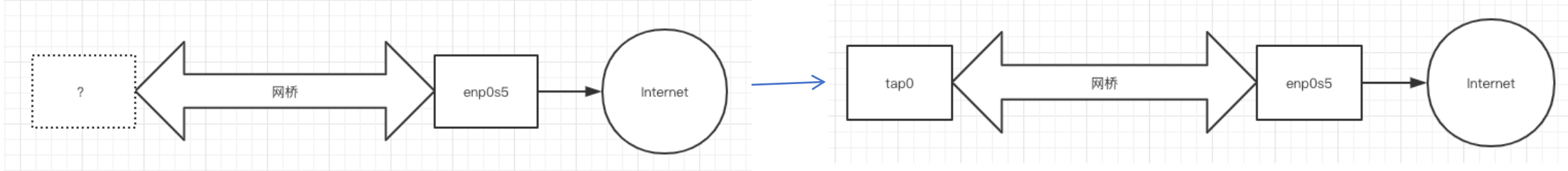
-net是一个遗留下来的选项。功能上和-device相似，用来定义虚拟机内部可见的设备参数，可以在虚拟机内部模拟一个网卡，如果定义了netdev就可以将这个网卡连接到netdev上，如果没有定义，就会默认连接到host主机的id为0的hub上，可以使用name={}参数来指定hub的名称。

-netdev user,id=net0,hostfwd=::2222-:22
-net nic,model=e1000,netdev=net0

不建议用这种方式。

```
Network devices:
name "e1000", bus PCI, alias "e1000-82540em", desc "Intel Gigabit Ethernet"
name "e1000-82544gc", bus PCI, desc "Intel Gigabit Ethernet"
name "e1000-82545em", bus PCI, desc "Intel Gigabit Ethernet"
name "e1000e", bus PCI, desc "Intel 82574L GbE Controller"
name "i82550", bus PCI, desc "Intel i82550 Ethernet"
name "i82551", bus PCI, desc "Intel i82551 Ethernet"
name "i82557a", bus PCI, desc "Intel i82557A Ethernet"
name "i82557b", bus PCI, desc "Intel i82557B Ethernet"
name "i82557c", bus PCI, desc "Intel i82557C Ethernet"
name "i82558a", bus PCI, desc "Intel i82558A Ethernet"
name "i82558b", bus PCI, desc "Intel i82558B Ethernet"
name "i82559a", bus PCI, desc "Intel i82559A Ethernet"
name "i82559b", bus PCI, desc "Intel i82559B Ethernet"
name "i82559c", bus PCI, desc "Intel i82559C Ethernet"
name "i82559er", bus PCI, desc "Intel i82559ER Ethernet"
name "i82562", bus PCI, desc "Intel i82562 Ethernet"
name "i82801", bus PCI, desc "Intel i82801 Ethernet"
name "ne2k_pci", bus PCI
name "pcnet", bus PCI
name "rocker", bus PCI, desc "Rocker Switch"
name "rtl8139", bus PCI
name "tulip", bus PCI
name "usb-net", bus usb-bus
name "virtio-net-device", bus virtio-bus
name "virtio-net-pci", bus PCI, alias "virtio-net"
name "virtio-net-pci-non-transitional", bus PCI
name "virtio-net-pci-transitional", bus PCI
name "vmxnet3", bus PCI, desc "VMWare Paravirtualized Ethernet v3"
```

网络配置的方式



2.tap/bridge模式

tap 属于 Linux 内核支持的一种虚拟化网络设备，还有 tun 也属于这种设备，它们完全由软件模拟实现，tun/tap 负责在内核协议栈和用户进程之间传送协议数据单元。tun 工作在网络层，而 tap 工作在数据链路层，tun 负责与应用程序交换 IP 数据包，而 tap 与应用程序交换以太网帧。所以 tun 经常涉及路由，而 tap 常用于网络桥接。

所谓桥接可以简单理解为在两张网卡之间搭一座桥，一端有数据就可以通过桥走到另一端，对于实现 QEMU 虚拟机通信正合适。

配置的方式比较繁琐，需要在host主机上先创建网桥，在飞书文档里面有一个相关的文档：
https://fwazw08p4j.feishu.cn/wiki/wikcnoGRRRIG8wmMKulhWlRS2Gc
另一个比较详细的博客可以参考：https://wzt.ac.cn/2021/05/28/QEMU-networking/

3.socket模式：在多个qemu虚拟机之间构建一个网络。

一个不容易发现的错误：如果添加端口映射之后报错，可以看一下是不是你的端口正在被使用（比如另一个终端的qemu没有关：）

共享宿主机的目录

* 重新编译qemu在configure中添加--enable-virtfs参数

* 在qemu虚拟机的kernel中添加9p的支持

* 还没有尝试

* 见[13] [14]

参考资料

[1] archlinux/wiki/QEMU一些说明详细的使用方法：https://wiki.archlinuxcn.org/wiki/QEMU
[2] Linux虚拟化的方式：https://www.cnblogs.com/LoyenWang/p/13510925.html
[3] qemu官方文档：https://www.qemu.org/docs/master/index.html
[4] archlinux的qemu参数手册：https://man.archlinux.org/man/qemu.1
[5] 使用qemu启动一个虚拟机：https://cloud.tencent.com/developer/article/1039421
[6] qemu-system-aarch64常用命令的参考：https://www.micoder.cc/blog/2429.html
[7] qemu tap网络模式的配置：https://wzt.ac.cn/2021/05/28/QEMU-networking/
[8] qemu常用参数：https://quard-star-tutorial.readthedocs.io/zh_CN/latest/ext3.html#drive
[9] -machine type=virt：https://www.qemu.org/docs/master/system/riscv/virt.html
[10] 制作ISO镜像：https://www.cnblogs.com/aWxvdmVseXc0/p/15553891.html
[11] 网络配置的官方文档：https://www.qemu.org/docs/master/system/devices/net.html
[12] 飞书配置tap网络的文档：https://fwazw08p4j.feishu.cn/wiki/wikcnoGRRRIG8wmMKulhWlRS2Gc
[13] 共享宿主机目录作为qemu虚拟机的主文件系统：https://blog.mozcp.com/use-host-directory-for-geust-rootfs/
[14] qemu使用9p virtio支持host和guest中共享目录：https://oskernellab.com/2018/09/30/2018/0930-0003-qemu_use_9pnet_virtio_fs_to_share_folder/
[15] Linux initial RAM disk (initrd) overview、制作initrd：https://developer.ibm.com/articles/l-initrd/
[16] qemu网络的参考文档：https://wiki.qemu.org/Documentation/Networking
[17] network socket：https://linuxsimba.github.io/qemu-tunnel-types
[18] network socket：https://john-millikin.com/improved-unix-socket-networking-in-qemu-7.2
[19] Finding your way through the QEMU parameter jungle：https://archive.fosdem.org/2018/schedule/event/vai_qemu_jungle/attachments/slides/2539/export/events/attachments/vai_qemu_jungle/slides/2539/qemu_cli_jungle.pdf
[20] 内核参数：https://www.kernel.org/doc/html/v4.14/admin-guide/kernel-parameters.html

参考资料

[21] 树莓派的sd卡分区命名和boot：https://github.com/raspberrypi/noobs/wiki/Standalone-partitioning-explained
[22] 安装一个ubuntu系统到虚拟磁盘文件上：https://juejin.cn/s/qemu%20create%20ubuntu%20image
[23] QEMU ELI5 — Part 6, UEFI, BIOS & OVMF：https://medium.com/@tunacici7/qemu-eli5-part-6-uefi-bios-ovmf-7919facf7e31
[24] How does a kernel mount the root partition?：https://unix.stackexchange.com/questions/9944/how-does-a-kernel-mount-the-root-partition
[25] device tree：http://www.wowotech.net/linux_kenrel/why-dt.html
[26] bootloader： https://www.scaler.com/topics/operating-system/what-is-a-bootloader/
[27] 一个启动的例子：https://blog.csdn.net/jiangwei0512/article/details/108176837