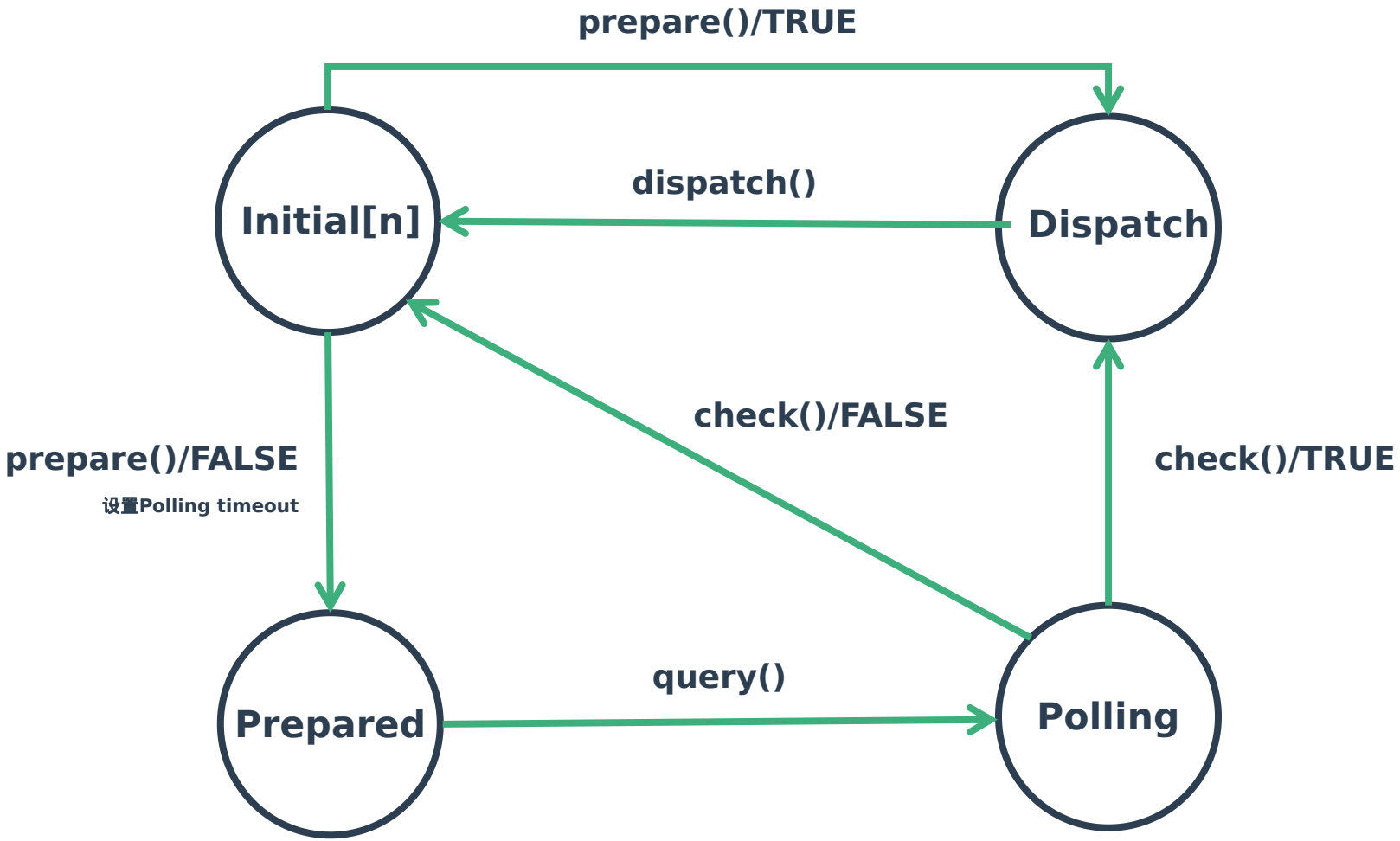
# qemu

# 线程模型

```
static QemuOptsList *vm_config_groups[48];
```

# GLIB事件循环机制

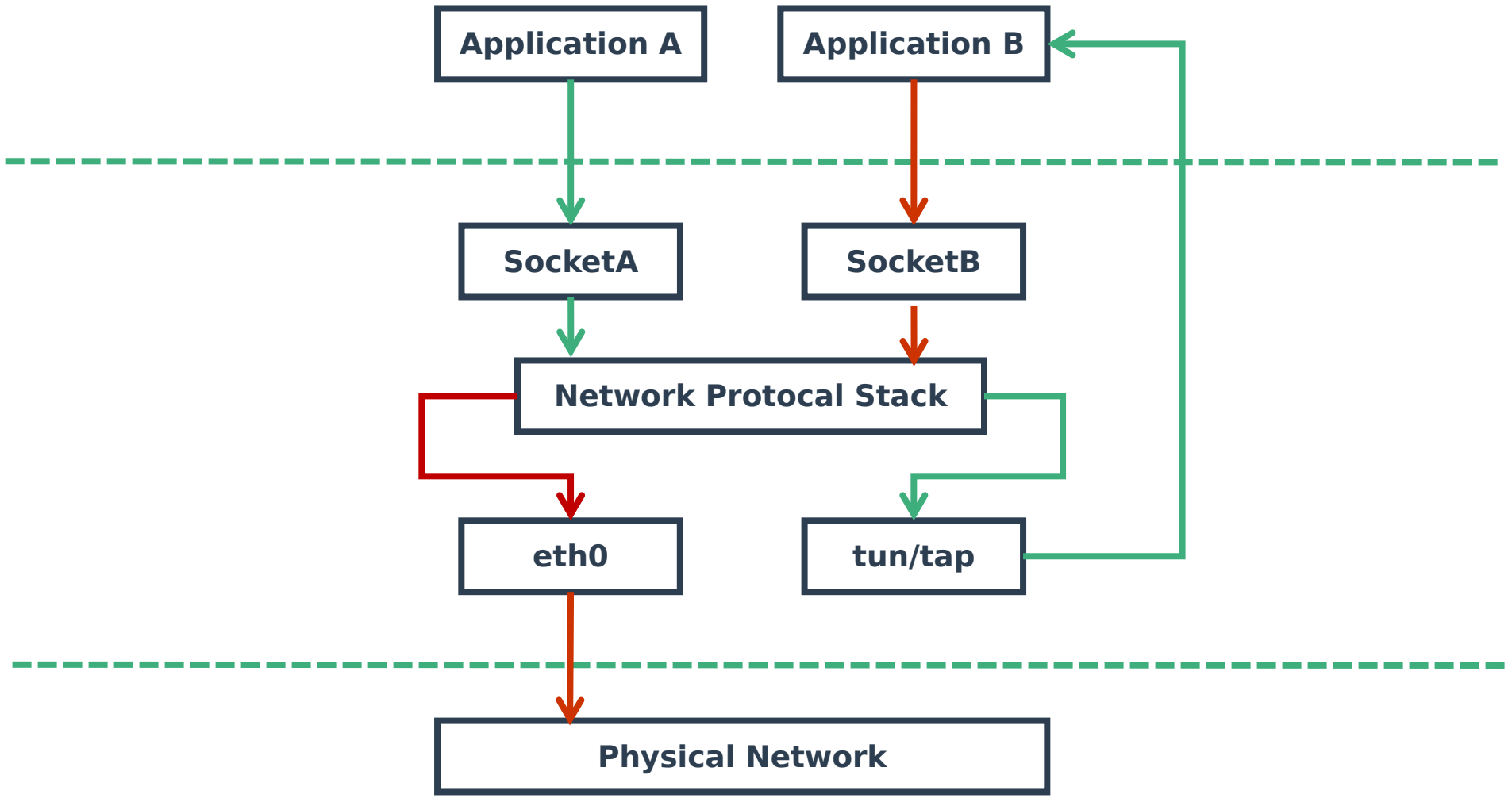
# GLIB事件循环机制状态图



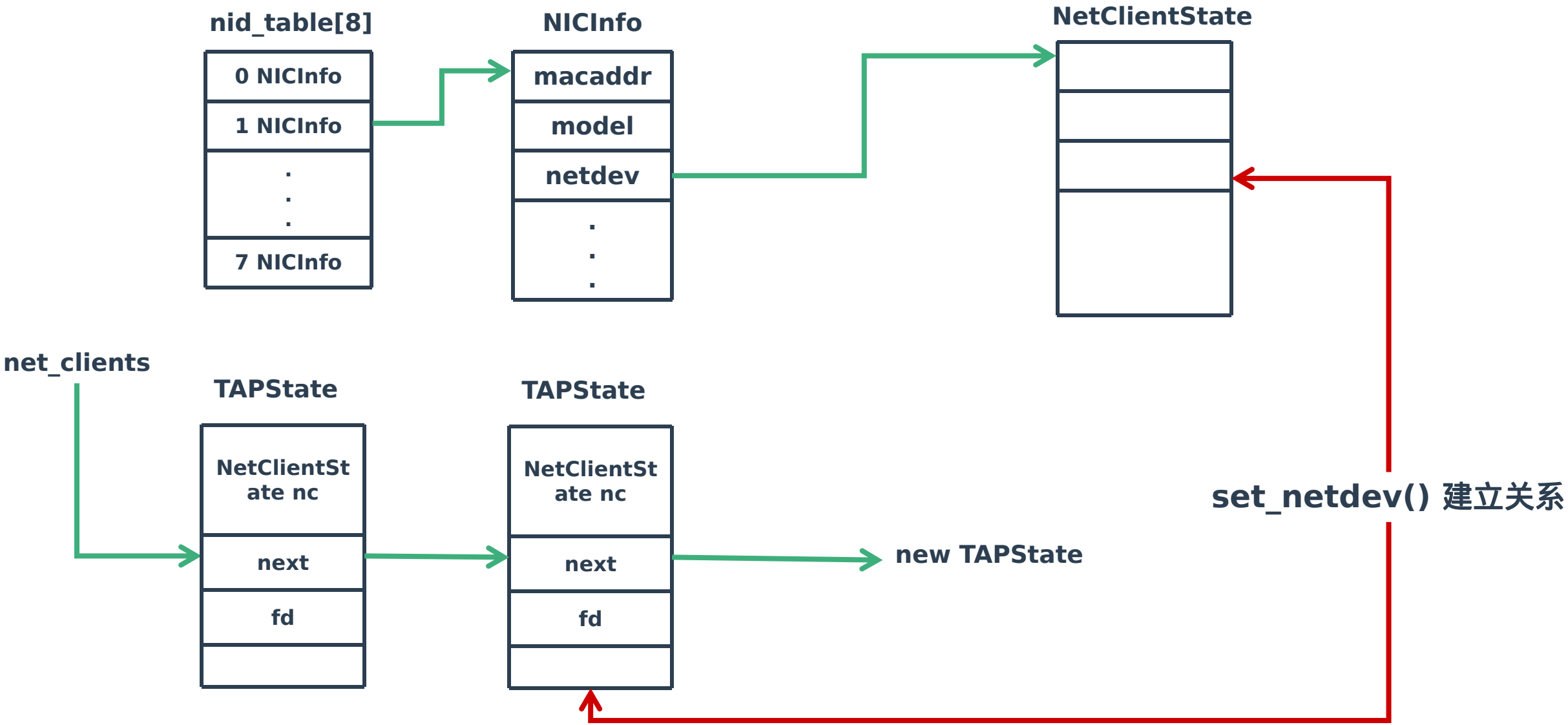
# QEMU内存


pc.ram
pc.bios
pc.rom
vga.vram
vga.rom
e100.rom
/rom@/etc/apci/table
/rom@etc/table-loader
/rom@etc/acpi/rdsp

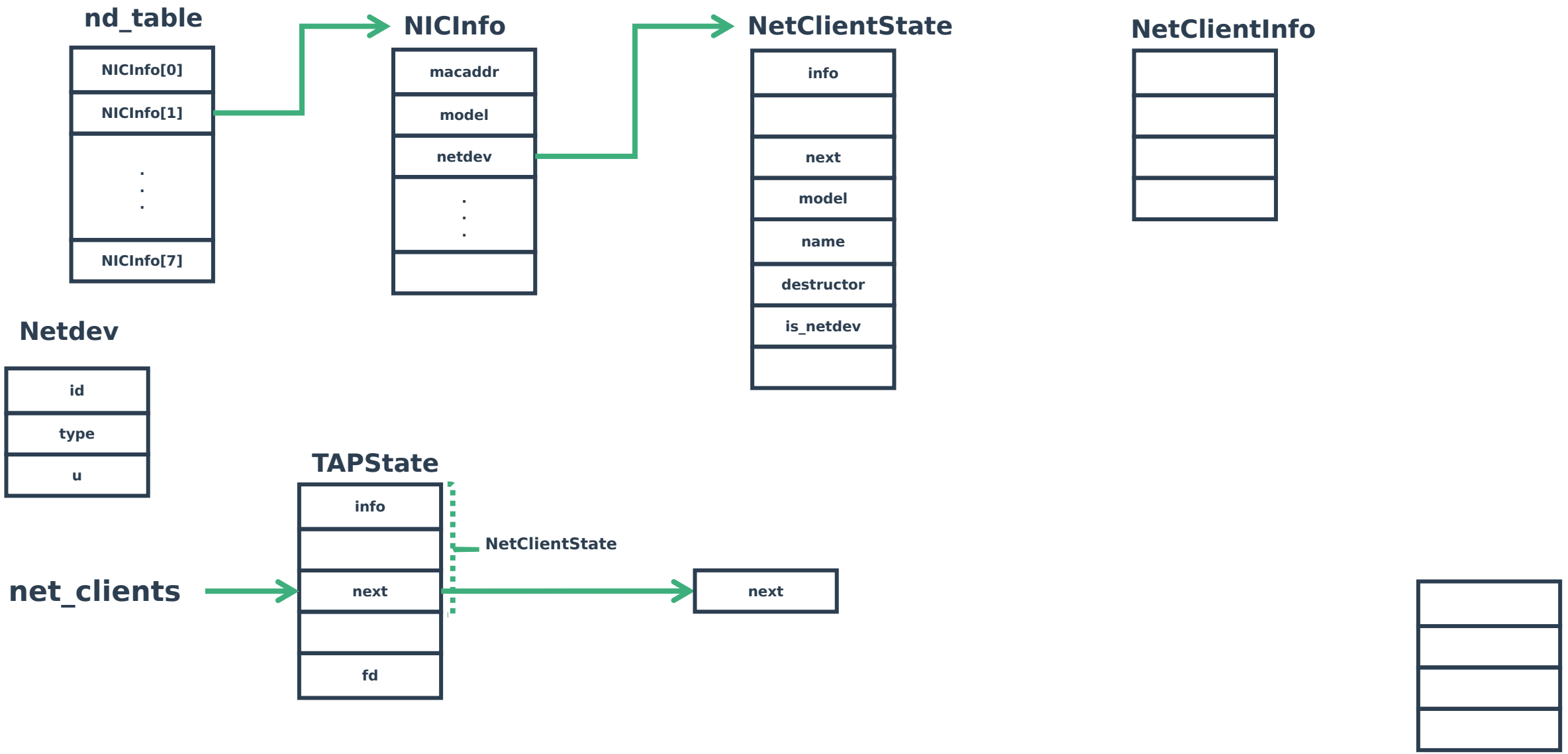
# QEMU TAP/TUN 原理

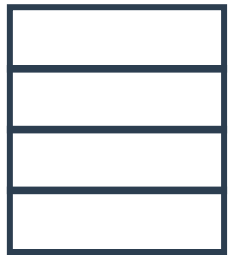


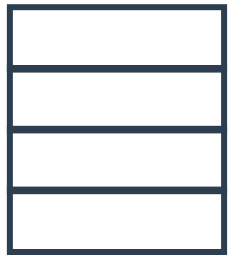
# QEMU TAP/TUN 实现



# 前后端设备


nd_table
NICInfo[0]
NICInfo[1]
NICInfo[7]
NICInfo
macaddr
model
netdev
NetClientState
info
next
model
name
destructor
is_netdev
NetClientInfo
Netdev
id
type
u
TAPState
info
NetClientState
net_clients
next
next
fd



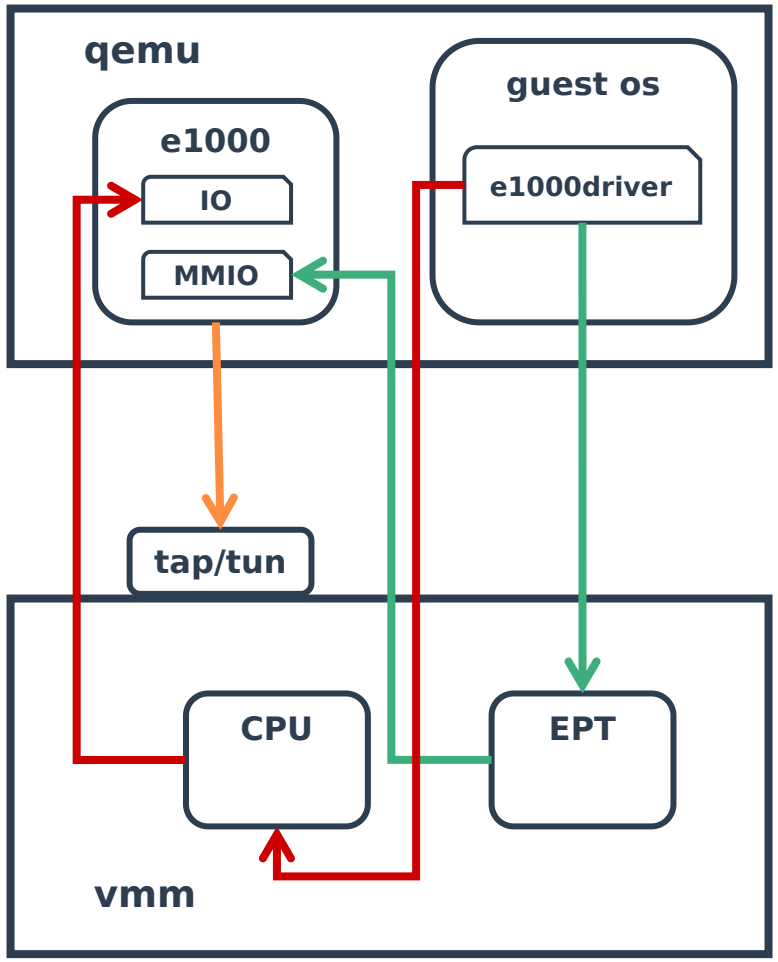


# virtio

本质上是内存虚拟化的应用

GUEST OS物理地址（GPA）可直接被QEMU的访问（HVA），因为在同一个进程中

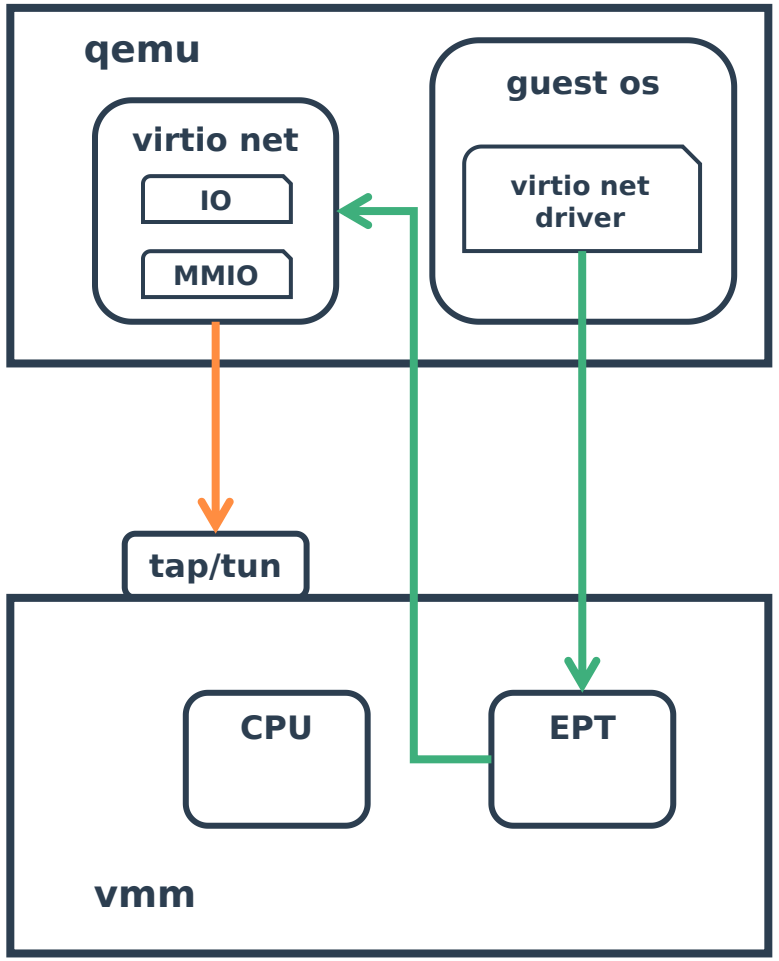
完全虚拟化




vm extit

no vm extit

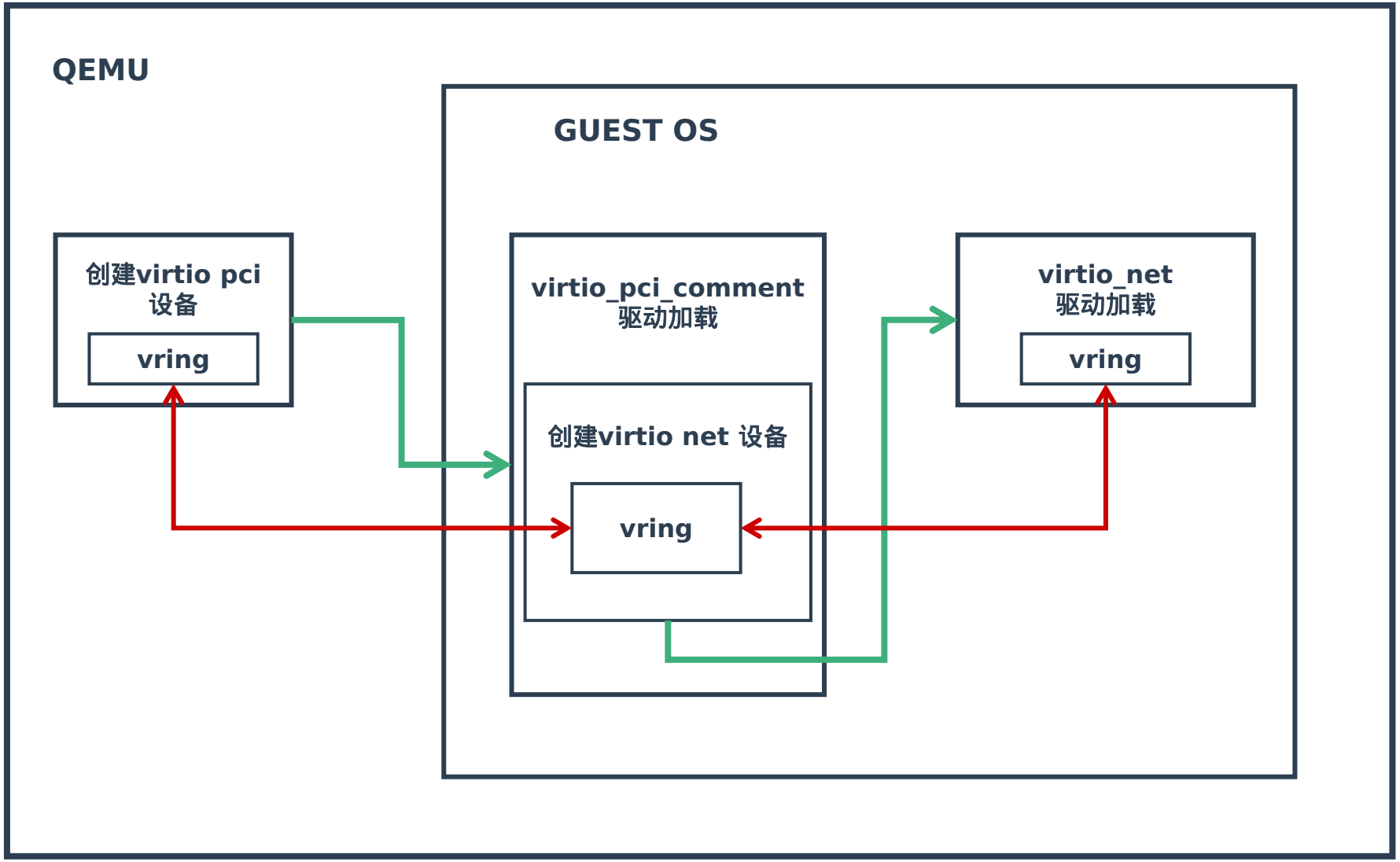
virtio虚拟化

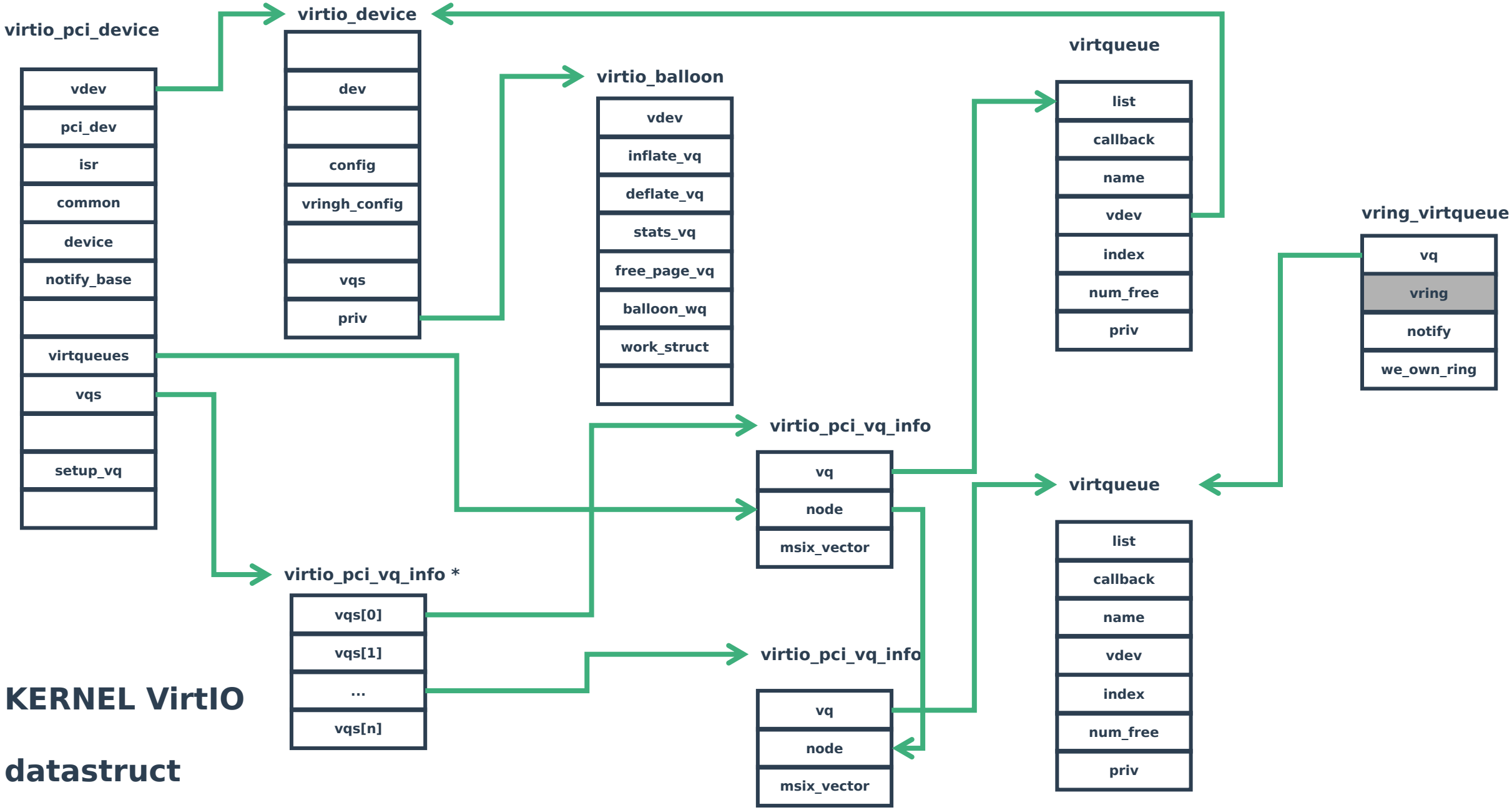


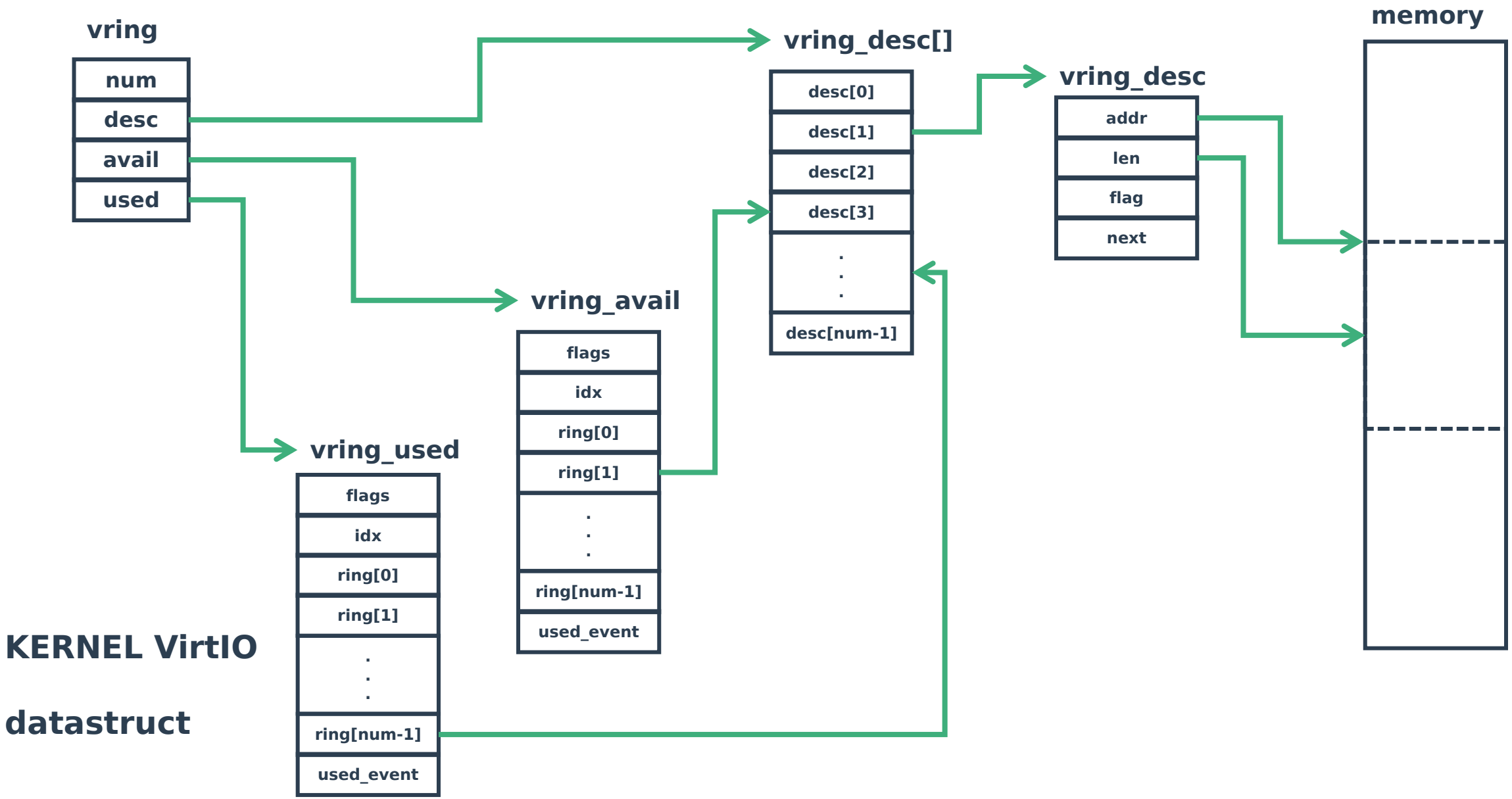

转化IO操作为内存操作

no vm extit

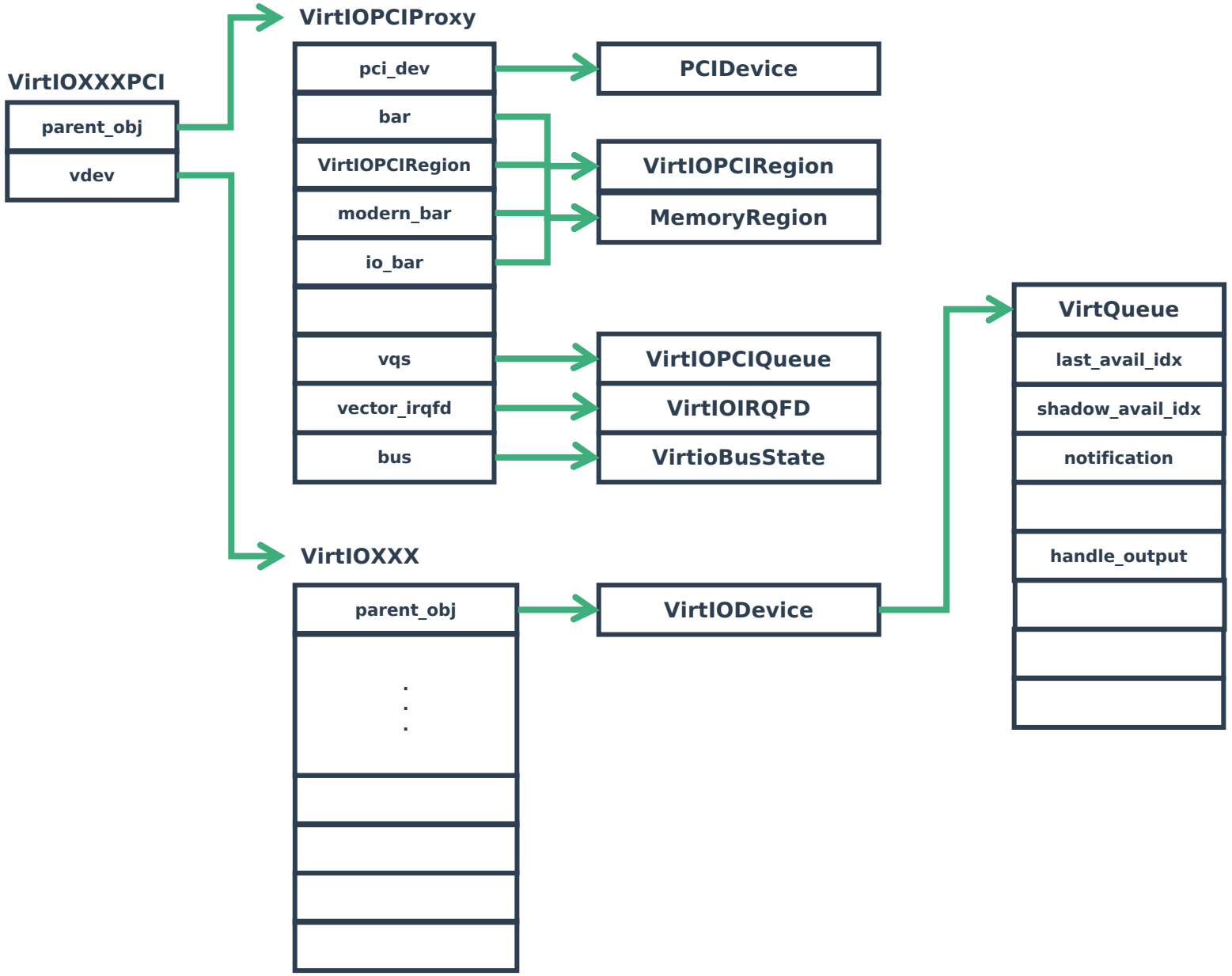
# VirtIO 设备


QEMU
GUEST OS
创建virtio pci
设备
vring
virtio_pci_comment
驱动加载
创建virtio net 设备
vring
virtio_net
驱动加载
vring


virtio_pci_device
vdev
pci_dev
isr
common
device
notify_base
virtqueues
vqs
setup_vq
virtio_device
dev
config
vringh_config
vqs
priv
virtio_balloon
vdev
inflate_vq
deflate_vq
stats_vq
free_page_vq
balloon_wq
work_struct
virtqueue
list
callback
name
vdev
index
num_free
priv
vring_virtqueue
vq
vring
notify
we_own_ring
virtio_pci_vq_info
vq
node
msix_vector
virtio_pci_vq_info *
vqs[0]
vqs[1]
...
vqs[n]
virtio_pci_vq_info
vq
node
msix_vector
virtqueue
list
callback
name
vdev
index
num_free
priv
KERNEL VirtIO
datastruct


vring
num
desc
avail
used
vring_desc[]
desc[0]
desc[1]
desc[2]
desc[3]
desc[num-1]
vring_desc
addr
len
flag
next
memory
vring_avail
flags
idx
ring[0]
ring[1]
ring[num-1]
used_event
vring_used
flags
idx
ring[0]
ring[1]
ring[num-1]
used_event
KERNEL VirtIO
datastruct

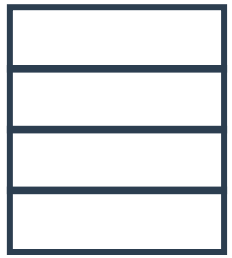
# QEMU VirtIO Device

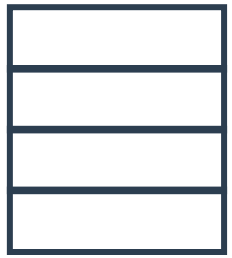


VirtQueue

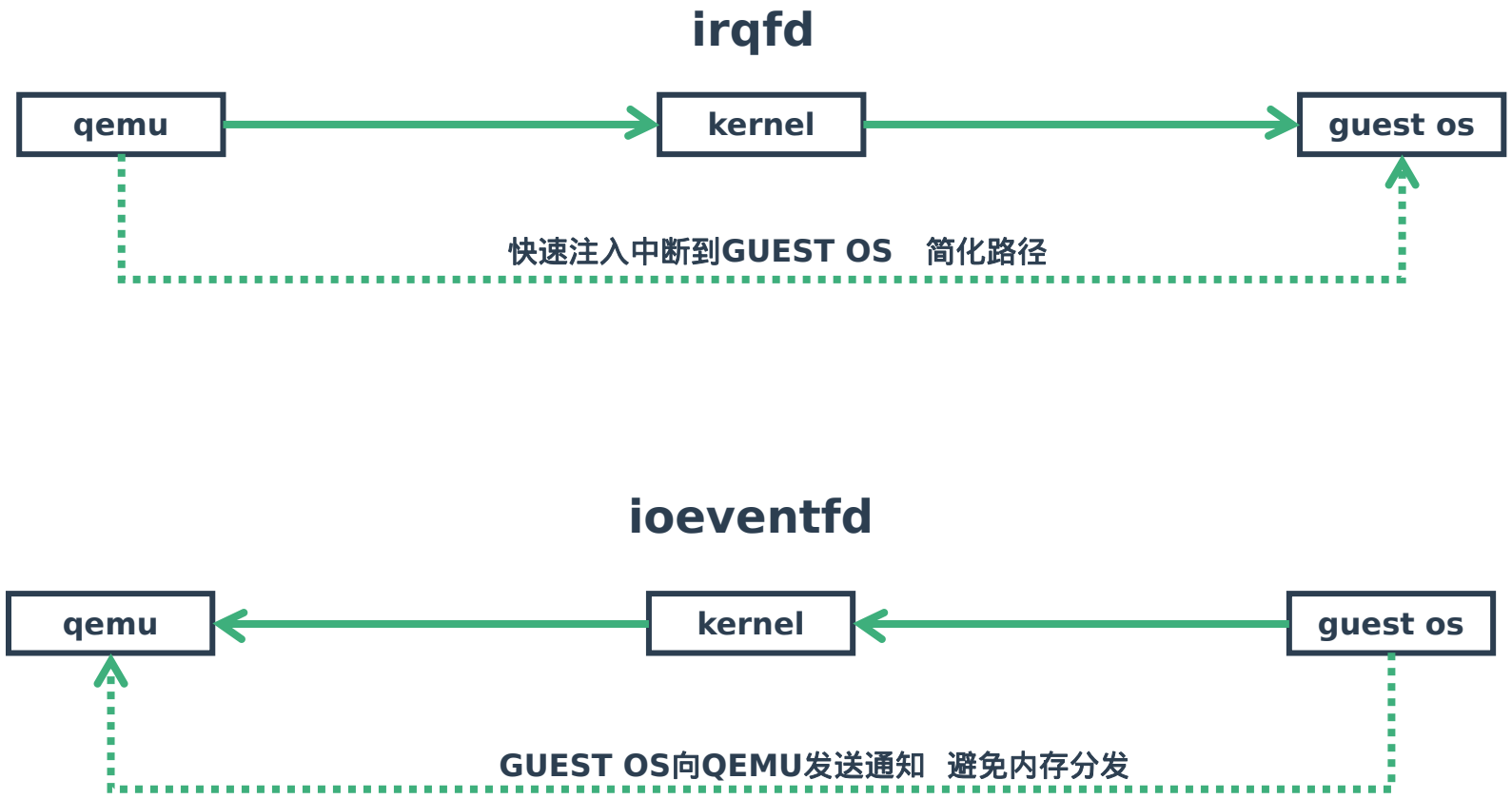
QEMU VirtIO

datastruct





# ioeventfd 与 irqfd



# vhost 框架