0. 基线（你已完成）

- 能 `duct0 at pci…` attach；`/dev/duct0` 可打开，骨架 `read/write/ioctl/kqfilter` 存根可用。
- 保持 `duct.c` 中**不调用** `cdevsw_add/cdevsw_del`；由 config 生成的 `cdev_duct_init(NDUCT,duct)` 管 cdevsw。
- GENRIC 里有 `duct* at pci?`；`files.pci` 里 `device/attach/file` 一条龙（PCI 驱动不是 pseudo-device）。

---

1. BAR 访问与信息打印（你已部分完成，可整理）

写什么

- 用 `bus_space_read_4` 读取 `VMAJ/VMIN/HWADDR` 打印出来。
- 只读 BAR0 的 0x00–0x50 区域，**不要**触碰 reserved。
  见*规范第 3 节 PCI 接口*—BAR0 寄存器图、偏移与含义（页 2 的表和图） a2-device-spec 。

怎么测

```sh
dmesg | grep duct      # 看 vX.Y 和 hwaddr 是否打印
doas ductctl -i        # GET_INFO 返回版本和地址
```

2. 事件与中断骨架（仅读清 EVFLAGS）

写什么

- 建立 MSI-X：vector0 事件、vector1 致命错误；handler 里先只 `rd4(EVFLAGS)` 读清（Read-to-clear）。
- 两条向量用途：0=事件，1=致命；见*规范第 4 节*（页 3） a2-device-spec 。
- 错误中断可先 `printf("%s: fatal FLAGS=0x%08x\n")`。

怎么测

- `attach` 后触发任何 ring 操作前读 EVFLAGS 不应报错；中断注册不 panic。

---

3. 先做命令环（比 TX/RX 简单） ✅ *老师 Hint 3 推荐*

写什么

- 分配一页连续内存作为 **CMDRING**（描述符 32B；ring 大小 2^n；OWNER=HOST 初值）。
- 写 BAR 的 `CMDBASE/CMDSHIFT`。
- 提交一条 **FLUSHFILT**（TYPE=5）命令：
  1. OWNER:HOST→填 TYPE=5 → store barrier → OWNER=DEVICE → barrier → DBELL=ring-index
  2. 等待 CMDCOMP 事件（EVFLAGS）或轮询 OWNER 回到 HOST
- 描述符格式与提交流程详见*规范 7.1/7.2*（页 9–12） a2-device-spec 。

怎么测

```sh
# 在 attach 的末尾就提交一次 FLUSHFILT
# 成功标准：EVFLAGS 出现 CMDCOMP；命令描述符 ERR=OK；OWNER 回 HOST
```

为什么先做命令环

TX/RX 还没搭好也能成功（START 例外），是最低成本验证 DMA ring/doorbell/中断整链条是否 OK 的方式（hint 强调先走 “FLUSHFILT/错误码” 路线）。

4. 加 START/STOP 命令（真正"运行/停止"）

写什么

- `open()` 里：若第一次打开 → 分配/初始化 **RX/TX ring**（见下一节）→ 写 `RXBASE/RXSHIFT`、`TXBASE/TXSHIFT`；读清 EVFLAGS；确认 FLAGS=0；再提交 **START**。
- `close()` 里：当最后一个 fd 关闭 → 提交 **STOP**；回收 RX/TX 资源（或延迟回收以复用）。
- START/STOP 的前置条件与错误码看*规范 7.3*（页 12–13）。不满足时设备会置 FLAGS 并停机，需要 reset（第 9 节） a2-device-spec 。

怎么测

```sh
doas mknod /dev/duct0 c 102 0
doas chmod 666 /dev/duct0
# 打开后应看到 START 成功；关闭后 STOP 成功（可在 dmesg 打印）
```

5. RX ring 路径（先做收包→read）

写什么

- 分配若干 DMA 缓冲，按描述符格式填写：`LENGTH1/POINTER1...`，OWNER=DEVICE；每次**补满**队头一段。
- 收到 **RXCOMP** 时：在 RX ring 上从上次消费位置开始扫，找到 OWNER=HOST 的条目，读取 `PKTLEN/SOURCE/DESTINATION`，拷贝入内核收包队列，OWNER 复位为 DEVICE 并回补；最后 `wakeup(&rxq)` 和 `knote()`。
- 描述符格式 / 初值要求 / 接收流程见*规范 6.1/6.2/6.4*（页 8–10） a2-device-spec 。
- 读路径实现：
  - 阻塞 read：队列空则 `msleep_nsec`；非阻塞 `EAGAIN`（*作业 4.2.3* 要求） comp3301-a2 。
  - 输出：**完整 Ductnet 头在首部**（SOURCE/DEST/LENGTH/RSVD），后跟载荷；支持 `readv`。

怎么测

```sh
# 先给设备加本机单播过滤器（见下一节），再用 echo 服务验证收包：
doas ductclient -e            # 发送到 0xEC60，应该收到原样数据（规范 4.4.1）:contentReference[oaici
```

6. 过滤器命令（ADDFILT/RMFILT/FLUSHFILT）与 ioctl

写什么

- ioctl `DUCTIOC_ADD_MCAST` / `RM_MCAST` → 在 **CMDRING** 填 `FILTMASK/FILTADDR` 字段并提交命令，阻塞直至 CMDCOMP（*作业 4.2.6.2/3* 的阻塞语义与 errno 约定） comp3301-a2 。
- 打开设备后，务必**至少**给 RX 加本机单播过滤器，否则不会收任何包（*规范 7.4*；并且 6.4 描述了收包仅在 OWNER=DEVICE 且匹配时发生） a2-device-spec 。
- 非多播时需 `EINVAL`； `ENOSPC/ENOENT` 按规范返回。

怎么测

```sh
doas ductctl -i           # 看地址
# 发空包到 QOTD（0xBEEF），应返回一条文本（规范 4.4.1）:contentReference[oaicite:9]{index=9}
doas ductclient -q
```

7. TX ring 路径（write）

写什么

- `write()` 校验 userland 头，复制 payload 到 DMA 缓冲（或零拷贝策略，建议先复制），填 `DESTINATION`、`LENGTH/POINTER`，OWNER=DEVICE → barrier → DBELL 高位+索引（*规范 6.3*） a2-device-spec 。
- 事件里扫 TX ring，OWNER 回 HOST 时释放缓冲并 `wakeup()`、`knote()`。
- 阻塞/非阻塞语义： ring 满时非阻塞 `EAGAIN`；阻塞 `msleep_nsec` 等待空间（*作业 4.2.4*） comp3301-a2 。

怎么测

```sh
# Echo/QOTD 都会触发真实发包
doas ductclient -e
doas ductclient -q
```

8. EVFLAGS 事件处理与 kqueue

写什么

- Vector0：读 `EVFLAGS`（read-to-clear）；根据 `TXCOMP/RXCOMP/CMDCOMP` 调用各自的 “完成扫”。
- Vector1：致命错误 → 打印 `FLAGS` 中错误，进入错误态（阻断 I/O 并返回 `EIO`）。
- kqueue：
  - `READ` ready：收包队列非空；
  - `WRITE` ready：TX ring 有空闲；
    作业 4.2.5 指定了事件的含义（W/R 两类） comp3301-a2 。

怎么测

```sh
# ductchat 使用 kqueue 的非阻塞 I/O，打开两个终端进入聊天室（0x8000CAFE）
ductchat   # 另一个 VM 或同一 VM 再开一个
```

---

9. 非阻塞 I/O 语义完善

- read 空 → `EAGAIN`；write 满 → `EAGAIN`；都可被信号中断（*作业 3.4/4.2.3/4.2.4 的非阻塞与信号语义*） comp3301-a2 。
- 提供 poll/select（OpenBSD 上由 kqueue 衍生实现）。

---

10. 并发与同步

- `sc_mtx` 保护 ring 索引、收包队列、过滤器列表等；中断 handler 与用户路径互斥。
- 支持**多进程/线程并发**（写序列化但无需严格有序；读可无序），这是评分项中 “Concurrency” 的重点（*作业 4.1.1*） comp3301-a2 。
- 所有内存屏障在写 OWNER 前后、扫 ring 前后要到位（规范在环节和流程段落多次强调 store/load barrier） a2-device-spec 。

11. 错误处理与复位（可选加分）

- FLAGS 任何位被置位 → 置设备停机、向用户返回 `EIO` 并 `printf` 错误（*规范 第 8 节*） a2-device-spec 。
- 可实现 reset（写 FLAGS=0x80000000 轮询到 0；*规范 第 9 节*） a2-device-spec 。

---

12. 收尾与鲁棒性

- 资源释放、错误路径回滚、长度/拷贝边界检查、锁整洁、assert/防御式编程（评分表的 Robustness 条目） assignment2-marks 。
- style(9) 规范、cstyle.pl 通过（Style 打分项） comp3301-a2 。

每步对应的“通过标准 & 命令”

阶段	通过标准	命令/工具
1	dmesg 打印 vX.Y + hwaddr	`dmesg
2	中断登记不 panic，读清 EVFLAGS	自测/printf
3	FLUSHFILT 提交→CMDCOMP 置位	在 attach 里提交；看 dmesg
4	open→START 成功；close→STOP 成功	打开/关闭 `/dev/duct0`
5	能收到 echo/QOTD 数据	`ductclient -e` / `-q`
6	ioctl 添加/移除过滤器，阻塞直至完成	`ductctl -a` / `-r`（参数合法性校验）
7	write 真正发包，TXCOMP 回来	`ductclient -e/-q`
8	kqueue 可用，ductchat 工作	`ductchat` 多端联动
9	非阻塞语义 EAGAIN 正确	`fcntl(O_NONBLOCK)` + 小脚本
10	并发压测不死锁	多进程/线程同时读写
11	FLAGS 错误能打印/返回 EIO（可选重置）	控制接口注入错误 / 自测
12	cstyle 通过；关闭资源不泄漏	`cstyle.pl`、反复开关/读写

小贴士（容易踩坑的点）

- **先命令环、后 TX/RX**：成本低、反馈快（老师 Hint 3 推荐做法）。
- **START 前置条件**：必须先写好 BASE/SHIFT、完成 RX/TX 初始 OWNER=HOST、读一次 EVFLAGS=0、FLAGS=0，否则会触发 SEQ/HWERR 并停机（规范 5、7.3） a2-device-spec 。
- **EVFLAGS 是 read-to-clear**：只读一次，后续事件要靠 ring 扫描补全（规范 6.5） a2-device-spec 。
- **内存屏障**：OWNER 写前后要 store barrier；扫描 ring 前要 load barrier（规范 6.3/6.5） a2-device-spec 。
- **非阻塞/阻塞**：按作业 4.2.3/4.2.4 的语义实现，支持被信号打断（EINTR）（*作业*） comp3301-a2 。

和打分项的对应（摘自 Marksheet）

- 功能：attach、字符设备、open/close/ioctl、ring 正确并发（12 分） assignment2-marks 。
- 鲁棒：错误处理、边界检查、并发锁、防御式编程（7 分） assignment2-marks 。
- 风格与反思：style(9)、reflection（6 分）。