



# 内核配置维护中的重复劳动与割裂

Coelacanthus

Arch Linux RISC-V

2024-07-14

# 自我介绍



Coelacanthus a.k.a. Celeste Liu

- Arch RISC-V 维护者
- AOSC 贡献者（虽然已经摸鱼很久）
- 复古计算机爱好者

# Linux 上游



- 每个架构有自己的 defconfig，由架构维护者维护
  defconfig: 预定义的配置集，编译的时候将 defconfig 作为基础，然后将未提及的选项设为默认值。
- 对于有些架构，不同的硬件有自己独立的 defconfig

# Debian



分层级的维护方式，有一个公共的 config，然后在这之上叠加具体的架构或者变体的配置。[1]

```
def merge(self, section, arch=None, featureset=None, flavour=None):
    ret = {}
    ret.update(self.get((section,), {}))
    if featureset:
        ret.update(self.get((section, None, featureset), {}))
    if arch:
        ret.update(self.get((section, arch), {}))
    if arch and featureset:
        ret.update(self.get((section, arch, featureset), {}))
    if arch and featureset and flavour:
        ret.update(self.get((section, arch, None, flavour), {}))
        ret.update(self.get((section, arch, featureset, flavour), {}))
    return ret
```

[1] python/debian_linux@kernel-team

# Arch Linux



每个内核分别维护一个完整的 config 文件，不同内核之间的同步靠维护者互相转达。

# Arch Linux RISC-V



维护一个 RISC-V 需要额外设置的配置项，并将其叠加在 Arch Linux 的内核配置之上。

# Fedora & AOSC OS



每个内核变体和架构分别维护完整的 config，相较于 Arch Linux 其不同变体的配置在同一个仓库中心化维护。

# NixOS



在上游 defconfig 的基础上应用 Nix 自己的配置。[2]

[2] linux/kernel/generate-config.pl@nixpkgs

# 厂商内核



大多直接维护一个完整的配置了事。

# 重复劳动



每个架构、每个发行版、每个设备、内核上游和发行版下游需要分别做出同一个修改。

# 割裂



由此衍生出的各发行版、各架构和各设备配置文件对同一个选项的配置不同步。

# 以 RT_GROUP_SCHED 为例



该选项对通用操作系统基本不可用，上游的许多架构或者硬件的 defconfig 已经关掉了这个选项，但是对于某些架构或硬件的 defconfig，以及某些发行版的内核配置中，仍然保留了该选项。[3]

当一个选项需要变更时，难以通知到众多的架构和发行版维护者，随着时间的推移，各架构、各发行版的配置呈现割裂的趋势，尽管很大一部分选项是所有人都认为应当开启或关闭的。

[3] https://lore.kernel.org/lkml/20240530111947.549474-8-CoelacanthusHex@gmail.com/T/

# 以 RT_GROUP_SCHED 为例



```
For cgroup v1, if turned on, and there's any cgroup in the "cpu" hierarchy it
needs an RT budget assigned, otherwise the processes in it will not be able to
get RT at all. The problem with RT group scheduling is that it requires the
budget assigned but there's no way we could assign a default budget, since the
values to assign are both upper and lower time limits, are absolute, and need to
be sum up to < 1 for each individal cgroup. That means we cannot really come up
with values that would work by default in the general case.[1]

For cgroup v2, it's almost unusable as well. If it turned on, the cpu controller
can only be enabled when all RT processes are in the root cgroup. But it will
lose the benefits of cgroup v2 if all RT process were placed in the same cgroup.
```

# 配置冲突



因为设计原理或者开发水平的原因，各方所需求的配置之间可能是互相冲突的。

# 以 PREEMPTION 与 FTRACE 在 riscv 上互斥为例



```
    select HAVE_FTRACE_MCOUNT_RECORD if !XIP_KERNEL
    select HAVE_FUNCTION_GRAPH_TRACER
    select HAVE_FUNCTION_GRAPH_RETVAL if HAVE_FUNCTION_GRAPH_TRACER
    select HAVE_FUNCTION_TRACER if !XIP_KERNEL && !PREEMPTION
    select HAVE_EBPF_JIT if MMU
    select HAVE_GUP_FAST if MMU
    select HAVE_FUNCTION_ARG_ACCESS_API
```

# 以 PREEMPTION 与 FTRACE 在 riscv 上互斥为例



> In RISCV, we must use an AUIPC + JALR pair to encode an immediate, forming a jump that jumps to an address over 4K. This may cause errors if we want to enable kernel preemption and remove dependency from patching code with stop_machine(). For example, if a task was switched out on auipc. And, if we changed the ftrace function before it was switched back, then it would jump to an address that has updated 11:0 bits mixing with previous XLEN:12 part.

```
p: patched area performed by dynamic ftrace
ftrace_prologue:
p|      REG_S   ra, -SZREG(sp)
p|      auipc   ra, 0x? ──────────────> preempted
                                            ...
                                  change ftrace function
                                            ...
p|      jalr    -?(ra) <──────────────  switched back
p|      REG_L   ra, -SZREG(sp)
func:
        xxx
        ret
```

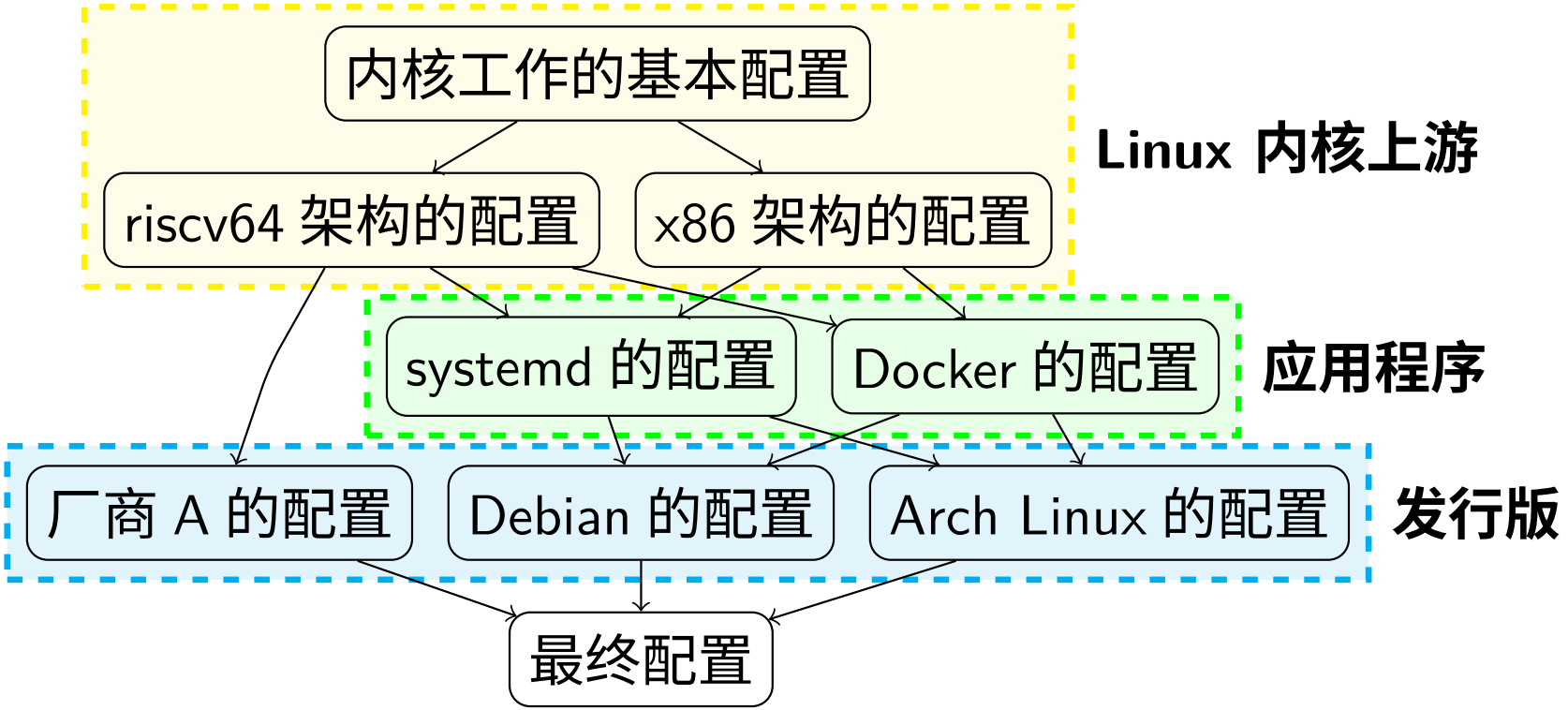
# 抛砖引玉



- 依赖关系和上下游关系决定了要分层次
- 各方有各方的需求决定了要分模块组合

# 抛砖引玉





# 需要的工具



- 组合配置的工具，能够处理选项依赖
- 检测配置组间选项冲突，给出有问题选项的 depgraph
- 能够给出当次生成和上次生成的变化概要
- 能够给出每一层的变化概要

# 依赖处理



本质上和包管理器的依赖求解是一样的，以参与组合的配置组明确指出的选项恒真为条件，求解 SAT。



# 谢谢大家！



# Q&A