AOSCC 2023 | 欢迎

AOSC 自动化展望

Ciel 与 GitHub Workflows

刘子兴 liushuyu011@gmail.com

杨欣辉 cyan@cyano.uk

Ciel 是个啥

- AOSC 打包工作特化容器管理器
  - 容器运行时 systemd-nspawn
  - 文件系统 overlayfs
  - 使用 D-Bus 和 systemd 沟通
  - 自带本地源功能

Ciel 工作流

- 初始化容器
- 刷新本地源
- 调用 acbs 打包
  - 下载源码
  - 安装依赖
  - 执行构建
- 清理容器

Ciel 设计决策

- 为什么使用 nspawn
- 为什么使用 overlayfs
- 为什么需要本地源设施

自动化第二环：GitHub Actions

- GitHub Actions 是 GitHub 的 CI 系统，具有工作流 (Workflows) 功能
- AOSC OS 的工作流主要依赖于 GitHub

为什么选择 GitHub Actions?

- AOSC OS 的开发工作流（主题制更新策略）依赖 PR
- Actions 能够涵盖 PR，从而实现 Topic 内自动打包
- 节省开发成本

总体设计

三类 Workflow:

- bleeding-edge{,nightly}: 前沿分支，每天自行检查更新
- in-topic-packaging: 发起 Topic 后自动构建 Topic 内的包
- after-topic-packaging: Topic 合并后自动打包

前沿分支：bleeding-edge

本 Workflow 分为更新检查及构建两个部分：

- 检查特定的软件包集合
  - 该集合涵盖 Core 分类的包，以及其他更新和维护简便的应用类软件包
  - 对于后者，如遇更新会自动发起 Topic
- 构建更新的软件包
  - 此步骤只适用于 Core 分类的软件包

Topic 相关: {in,after}-topic-packaging

- 只做两件事：打包和推包
- 打包的范围是 Topic 中注明的软件包列表
  - 因此要求各位贡献者注明构建顺序，必要时也需要添加反向依赖
  - 涵盖的架构是 amd64 和 arm64（包括 optenv32 和 noarch）
- 构建不通过时，禁止合并 PR
- 推包即推到 Topic 中

规划

- Runner 调用若干对应的脚本或程序，各自实现各自部分的功能
- 打包过程最终会调用 Ciel
- 自动打包仅在 amd64 和 arm64 上试点
- 推包时使用的 Credentials 储存在 Secret storage 中，供 Actions 使用

Artifacts

- Topic 相关的 Workflows 只上传构建日志
- 前沿分支的 Workflows 上传构建的包和日志
- 无论构建失败与否，都要上传

长期计划

- 最终自行造轮子，摆脱 GitHub Actions 的服务
- 同时扩展到绝大部分 Tier 2 架构

Ciel 自动化修改

- Ciel 当前主要面向人类用户
- systemd-nspawn 控制性不佳
- 没有自动化日志收集

Ciel 更可控的存在

- 多个运行时后端
- 多个文件系统后端
- 和 acbs/autobuild 更紧密的结合

Q&A