Linux 201 文档计划与杂谈

Keyu Tao, 2024/08/17

Part I: Linux 201

- https://201.ustclug.org
- https://github.com/ustclug/Linux201-docs

起源

- Linux 101 项目 (https://101.lug.ustc.edu.cn)
  - 2018 年开始在每年的春季学期开活动介绍 Linux 基础知识（只有 Slides）
  - 2020 年寒假开始编写 Linux 101 的文档
    - 第一个 commit 在 2020/1/31
    - COVID-19 => 大半年待家里 => 能腾出大量闲置的时间写东西
  - 目前基本完工，处于维护状态
- Then?
  - 从「用户」到「高级用户」（Power User）之间的 gap
  - 以年为单位实践与踩坑
  - 有足够经验 且 愿意加入维护的同学属于稀缺资源
  - 可以把这些经验整理出来吗?

起源 (cont.)

- 2021 年底开了一个新仓库
  - 当时还叫 "Linux 102"，作为 Linux 101 的后续
  - 然后……直到 2023 年 9 月开始才有实质的更新
- 开始更新最直接的原因：接手技术设施的同学实力不足，进度推进极其缓慢
  - 缺少足够的前置知识，技术文档不够用
  - 完全无法融入技术交流
- 即使有相对较好的基础知识，接手任务仍然不容易
  - 从「高级用户」到「系统管理员」之间仍然有一定的 gap
    - 例子：IPMI 是什么？SFF/LFF 是什么？
  - 学生运维的小规模 => 每个成员都需要对各个方向有基本的了解（某种程度的全栈？）
- 需要类似于 Linux 101 的文档项目 => Linux 201
  - 不是「内部技术文档」：这些知识应该是通用的
  - 系统介绍需要了解的知识与技能，提供实践的方案，以及经验与教训

设计

- 三个大板块
  - 运维（Ops）：包括基础的运维需要了解的内容，例　服务、日志管理，存储系统，网络等
  - 开发（Dev）：为快速入门，调试与魔改现有程序而设计的介绍性内容
  - 高级/选读内容：成熟度不足，或者不便于分类到上述两者的内容
- 设计原则/前置要求
  - Debian First
  - Linux 101 + C & Python 基础
- 计划结构展示

亮点

- 把底层逻辑搞清楚
  - 例子：NFS 的「子树检查」选项
- 我们拥有的独特经验
  - 例子：LVM 部分的 lvmcache（虽然现在我们不用了）

进度

- tl;dr: 进展很慢
  - 相比于 Linux 101，编写的门槛更高，也不再有大量窝在家里的时间条件
  - 完成度展示

欢迎加入！

- Linux 201 **不是**仅限于科大学生或者 USTCLUG 成员编写的项目
  - 果你想写作某一部分，欢迎联系！
  - 也欢迎捉虫！
- 目标：为学生技术社团带来实质的知识帮助
  - 可以直接给新人扔 201 的网站链接

Part II: 杂谈

- aka「我造的镜像站相关的小工具介绍」
  - 同步工具
  - 运维工具

tsumugu

- 镜像站有的时候无法用 rsync 同步上游内容
  - aptsync/yumsync 只适用于 APT/YUM 仓库
  - lftp/rclone 每个文件都会 HEAD，并且某些站点无法同步
- 2020 年尝试以 docker-ce 为目标写了一些 Go 代码，弃坑了
- 2023/7 重新开写
  - 主要的 motivation：proxmox 用 lftp 要超过 10 个小时才能同步完，上游拒绝开 rsync
  - 解决方法：parse HTML 来判断文件要不要同步（而不是每个都去 HEAD 一遍）
  - Rust（强类型偏好 + 作为语言练习）
  - 由于可以自己写 parser，因此 lftp/rclone 同步困难的页面也可以同步
  - 其他疑难问题（比　说 docker-ce 的「特性」）也可以缓解
- 未来工作
  - 处理 APT/YUM 仓库部分同步的不一致问题

yukina

- 巨大且利用率低的仓库的 caching 方案
  - nix-channels（只同步近期数日的包也需要近 5T）
  - pypi（bandersnatch 只增不减，~20T）
- OSPP 2021 的尝试：Rust 编写的 caching server
  - Thanks to @SeanChao!
  - 但是因为种种原因没有用上😢
- 2024/3 开写
  - 思路：从 Nginx log 了解文件的「流行程度」，在大小约束下下载流行的文件，删除过气的文件
  - 不需要繁琐的 web server 配置，也不需要维护一套独立的系统
  - 最新支持只 gc 不下载，因此（可能）可以和 `proxy_store` 一起用
- 统计
  - 已经为 nix-channels 部署，512G 限制下能够做到 ~80% 命中率
  - PyPI 命中率 ~85%

shadowmire

- 高校镜像站管理员的共识：bandersnatch 很烂，但是不得不用
  - sync 的时候不会删除上游没了的包（供应链攻击问题！）
  - verify 很慢很慢（每个包都要完整更新一遍）
  - 无法为下游提供同步（需要 XML-RPC API，镜像站不可能实现）
  - 不一致性问题（有些包没有 JSON API 文件，有些包没有 `index.v1_json` / `index.v1_html`）
- 观察
  - PyPI 的 XML-RPC API 的 `list_packages_with_serial()` 方法，可以返回所有包以及最后更新的 serial（可以看作是某种时间计数器）
  - 本地维护一个轻量的数据库（`import sqlite3`）
- 那就自己写一个吧!
  - 目前 LOC 1k，比 bandersnatch 小一个数量级
  - 支持只同步 index，因此可以和 yukina 一起用
- 与 TUNA（@Harry-Chen）合作验证

ayano

- 为镜像站特化的轻量级 Nginx log analyzer
  - follow access.log 文件
  - 筛选大请求
  - 按 /24 或者 /48 统计总大小
  - 不处理「大量小请求」的情况——其他的方法已经可以有效处理这种情况了
- 最新：对 daemon mode 的支持
  - 输出日志，可以喂给 fail2ban 动态 ban IP

chitose

- 和 iftop 干的事情类似，但是可以按 /24 或者 /48 归类统计流量
- `sudo pacman -S chitose`

admirror-speedtest

- USTC Mirrors 有多个不同 ISP 的出口 IP
  - 在编写同步任务配置的时候怎么知道哪个出口最快?
- 测速方法
  - 以不同的 IP 配置启动 rsync/curl/wget/git
  - 测试 3 轮，每个测试 30 秒
- 有趣的细节
  - 发现了 rsync 的一个 bug，会导致 rsync 无法正确 kill 自己的子进程
  - git 不支持修改 `bind()` 的 IP
    - 那就 `LD_PRELOAD` 魔改：libbinder
    - 也了解到了 seccomp user notification（但是有点麻烦，就没用）
    - 这也是 Hackergame 2023 的题目 `为什么要打开 /flag` 😡 的思路来源
- 未来工作
  - 支持 Docker network?

gitkeeper

- 服务器有一些配置以 git 仓库的配置存在
  - 但是很多时候改了之后忘了 commit/push
  - `git` 在某个 CVE 之后，加强了权限检查
    - 导致每次都要 `sudo -u someuser git -c user.name=example -c user.email=someuser@example.com commit ...`
    - 更没人愿意 commit 了
- gitkeeper 是轻量、系统级的 git 配置仓库管理器
  - 会帮你 `sudo`，帮你填上正确的 `user.name` 和 `user.email`，帮你对不是你的用户的仓库也用上你自己的 `.gitconfig`
  - `gitkeeper status` 一条命令查看系统仓库状态
  - `gitkeeper diff` / `gitkeeper commit` / `gitkeeper update`
  - `alias gitk="gitkeeper vcs ."`
    - 然后就像用 `git` 一样用 `gitk`

Q & A?