lintestor：RISC-V 软件包支持情况矩阵自动化测试工具

报告内容

1. lintestor 简介
2. 功能特性 & 技术实现
3. 当前支持的测试
4. 使用示例
5. 遇到的挑战
6. 未来计划

lintestor 简介

- 基于 Rust 的自动化测试系统
- 设计上支持多发行版，目前主要针对 Debian 软件包（Bianbu 实机测试尝试中）
- 目标：提供一个类似 tarsier-meta/report/info.md 的可用性矩阵
- 255doesnotexist/lintestor

必要性

- 各发行版软件包在 RISC-V 环境下的可用性能提供一个基本了解
- 希望能接续 isrc-cas/tarsier-meta 中的包可用状态部分
- 希望自动化 RISC-V 平台上的不同发行版的不同软件包的测试流程，直接提供大概可用性的一个矩阵

功能特性

1. 支持多发行版（设计上支持，目前主要是 Debian）
2. 自动管理 RISC-V 测试环境（QEMU 虚拟机启停）
3. 生成 Markdown 格式的测试结果矩阵
4. 支持跳过特定包的测试
5. 本地测试选项（`--locally` 参数）

[来源：week3.md, 主仓库代码部分第1-4点; week0.md, lintestor 主仓库代码部分第1点]

结果矩阵示例

软件包	种类	debian
apache	Web Server	✅ apache-2.4.62-1
clang	Compiler	✅ clang-version
docker		❓
erlang	Programming Language	✅ erlang-1:25.3.2.12+dfsg-1
gcc	Toolchain	✅ gcc-14.2.0
gdb	Debugger	✅ gdb-15.1-1

- ……

单软件包测试结果格式


```
{
    "distro": "debian",
    "os_version": "Linux version 6.9.9-riscv64 (debian-kernel@lists.debian.org) (riscv64-linux-gnu-gcc-13 (Debian 13.3.0-1) \
13.3.0, GNU ld (GNU Binutils for Debian) 2.42.50.20240710) #1 SMP Debian 6.9.9-1 (2024-07-13)",
    "kernel_version": "6.9.9-riscv64",
    "package_name": "apache",
    "package_type": "Web Server",
    "package_version": "2.4.62-1",
    "test_results": [
        {
            "test_name": "Apache Service Test",
            "passed": true
        }
    ],
    "all_tests_passed": true
}
```

技术实现

- 配置管理：使用 TOML 格式作为配置

```toml
distros = ["debian"]
packages = [
    "apache", "clang", "cmake", "docker", "erlang", "gcc", "gdb", "golang", "haproxy", "libmemcached", "lighttpd", "llvm", "mariadb", "nginx", "nodejs", "numpy", "ocaml",
    "openjdk", "perl", "python", "ruby", "rust", "sqlite", "varnish", "openssl", "postgresql", "redis", "runc", "scipy", "squid", "zookeeper"
]

startup_script = "./debian/start_qemu.sh"
stop_script = "./debian/stop_qemu.sh"
skip_packages = ["docker"]

[connection]
method = "ssh"
ip = "localhost"
port = 2222
username = "root"
password = "root"
```

- 通过 SSH 协议配合 SCP 实现远程（其实就是到 RISC-V 的 QEMU XD）执行测试及结果同步
- 利用 GitHub Actions，使用自托管的 Action runner，每周一自动化跑一次（还得调调）

当前可以尝试运行的的测试

发行版:

- Debian

软件包:

- Web 相关: Apache, Nginx, Lighttpd, HAProxy
- 解释器: Python, Ruby, Perl, Go, Rust, Erlang, OCaml, Node.js
- 数据库: MariaDB, PostgreSQL, SQLite, Redis
- 编译工具链: GCC, Clang, LLVM, CMake, GDB
- 容器相关: Docker, runc
- 科学计算的 Python 包: NumPy, SciPy
- 其他一些包: OpenJDK, OpenSSL, Varnish, Squid, ZooKeeper, libmemcached

总计: 1 个发行版，32 个软件包

使用示例

```
# 在本地运行测试
./lintestor --test --locally

# 使用 SSH 连接 QEMU 测试
./lintestor --test --aggr --summ
```

利用 TestRunner trait 机制实现不同行为代码易复用

```rust
pub trait TestRunner {
    fn run_test(&self, distro: &str, package: &str) -> Result<(), Box<dyn std::error::Error>>;
}
```

- RemoteTestRunner: 通过 SSH 调度测试任务并执行
- LocalTestRunner: 直接执行测试脚本

GitHub Actions 自动化 CI 测试

- 某一次执行...
- test-results.zip 打包了本次测试结果，包含以下文件：
  - reports.json （聚合后的测试结果）
  - summary.md （结果矩阵）
- 需要注意的是随着测试数量增加或意外 bug 可能会耗尽每月 Actions 时长。现通过在 Infra 上自建 Actions runner 解决。

遇到的挑战

- SSH 通信问题，wait_eof 的处理不对导致意外的结果
- 复杂软件包测试可能有复杂的依赖关系，可能考虑测试后复位镜像防止包之间的依赖关系相互干扰
- 依赖安装时 apt 可能会交互式询问是否继续，而忽略 -y 选项，因此 lintestor 在测试机（QEMU）上现场自动安装测试依赖时需要设置非交互式安装环境变量
- 虚拟环境与真实环境的差异，暂无法完美解决，可以在真实板子上跑 lintestor 的本地测试，冻结其结果至现有矩阵，定期手动更新

遇到的挑战

- 部分软件包完整 autopkgtest 测试项庞杂、笨重费时，又有部分软件包并未附带 autopkgtest，且有发行版局限性，最终未采用此种方式测试
- 图形化测试编写、校验有困难
- 测试结果可能不够健壮

未来计划

1. 在 BPI-F3 上部署 RISC-V GitHub runner [WIP]
2. 支持 Bianbu 的自动化测试 [WIP]
3. 探索支持 VNC 或 KVM 的图形化测试
4. 也许 GitHub Actions 测完应该提个自动 pr？待研究。
5. 扩展测试范围

参考外链

- isrc-cas/tarsier-meta:report/info.md: Linux Test Project (LTP)流行 Linux 发行版 for RISC-V 的应用支持情况测评和对比