SIG-reproducible-builds Team Gathering Schedule

16:00~16:10　可重复构建sig组建设背景、2022年openEuler可重复构建总体目标 --刘波

16:10~16:20　可重复构建洞察、标杆开源社区如何做可重复构建 -- 徐亮

16:20~16:30　openEuler当前TOP问题、什么会影响可重复构建　-- 徐亮

16:30~16:40　可重复构建sig组运作机制、如何推动上游社区 -- 吴峰光

16:40~16:50　可重复构建工程技术方案与工具链方案 -- 刘路

16:50~17:00　开放讨论

可重复构建sig组的建设背景、与2022年规划

开源社区工具链架构师

刘波

目录：

2022年软件供应链安全提到前所未有高度、如何证明?

OpenEuler

可重复构建（ Reproducible Builds）是证明软件供应链安全的必要手段

当前已纳入supplychainsecuritycon的topics

https://events.linuxfoundation.org/open-source-summit-north-america/about/supplychainsecuritycon/

Click above to submit a proposal to speak at SupplyChainSecurityCon, or one of the other Open Source Summit North America conferences.

SupplyChainSecurityCon topics include:

- Measuring Risk of Potential & Already-included OSS
- Countering Source Code Level Problems
  - Reducing the Likelihood of Vulnerabilities (e.g., Eliminating Entire Classes)
  - Countering Subverted Source Code Control Systems
- Countering Build Threats
  - Simplifying Verified Reproducible Builds
  - Ensuring Safe Transition from Source Code Control to Build System
  - Countering Compromised Build System
  - Countering Bypassed CI/CD
  - Countering Subverted Package Repository
  - Countering Use of Bad Package
- Countering Dependency Threats
  - Countering Use of a Bad Dependency
- Ensuring Users Know, With Confidence, What Software Components (at All Tiers) are Included

SUPPLYCHAIN SECURITYCON

BROUGHT TO THE COMMUNITY BY

THE LINUX FOUNDATION · OpenSSF · CLOUD NATIVE COMPUTING FOUNDATION · SPDX · ACT Automating Compliance Tooling · OPENCHAIN

The idea of a verified reproducible build is gaining traction. In such a build, the code can be verified as containing only code that came from the original source code.

"That means your build is designed so it will produce the same bits every time given the same source code," says Wheeler, who recently wrote a blog post on the subject for the Linux Foundation.

Wheeler says that most software now is not designed to be reproducible, but the Linux Foundation has funded some projects for reproducible builds. A new Linux Foundation project, the Open Source Security Foundation, is discussing whether to take on reproducible builds as a project.

2022年可重复构建SIG成立背景、工作目标、范围

OpenEuler

SIG名称

可重复构建SIG (reproducible-builds)

概述

可重复构建也被称为确定性编译，是一个编译软件的过程，目标是确保在使用相同的输入时（源代码、工具链、环境变量等）生成的二进制代码可以比特及重现。

Reproducible-Builds SIG组工作目标和范围

工作目标

- 在 openEuler 社区建设RPM体系可重复构建能力; 任意发布RPM包都可还原其源码、构建环境、依赖、构建工程配置等、且再次构建二进制比特位100%一致
- 回合工作成果、并推动上游社区的包达成可重复构建
- 基于核心包、外围包分阶段达成; 对齐Debian社区可重复构建能力

工作范围

- 基于 openEuler Release包Policy规范、对影响可重复构建因素进行约束
- 加入 https://reproducible-builds.org/ 组织、共享共建复用社区已有能力、并得到国际社区认可
- 基于 openEuler 社区建设可重复构建工具链
- 建立修复问题、回合上游社区的体系
- 建立与其他sig 组协同运作机制

https://gitee.com/openeuler/community/tree/master/sig/sig-reproducible-builds

可重复构建目标：通过2年建立可追溯、可重复构建能力、持平领先社区Debian 95%

OpenEuler

GAP

- **领先社区已建设10+年:** Debian 30192个包94.8%、 openSUSE 13929 个包 95.9%、NixOS 1572个core包 99.87%、openEuler可重复比例： 1500核心软件包65%、所有8000+软件包62%
- **推动上游社区软件修复工作量大：** 上游社区源码难运作，修复周期长（定位问题->提交issue->接受issue->提交PR修复->验证合入）

目标

建设社区可重复构建工具链，推动社区2022达成openGauss,MindSpore ,openEuler 1500 core包95%可重复，2023年所有包 98%可重复

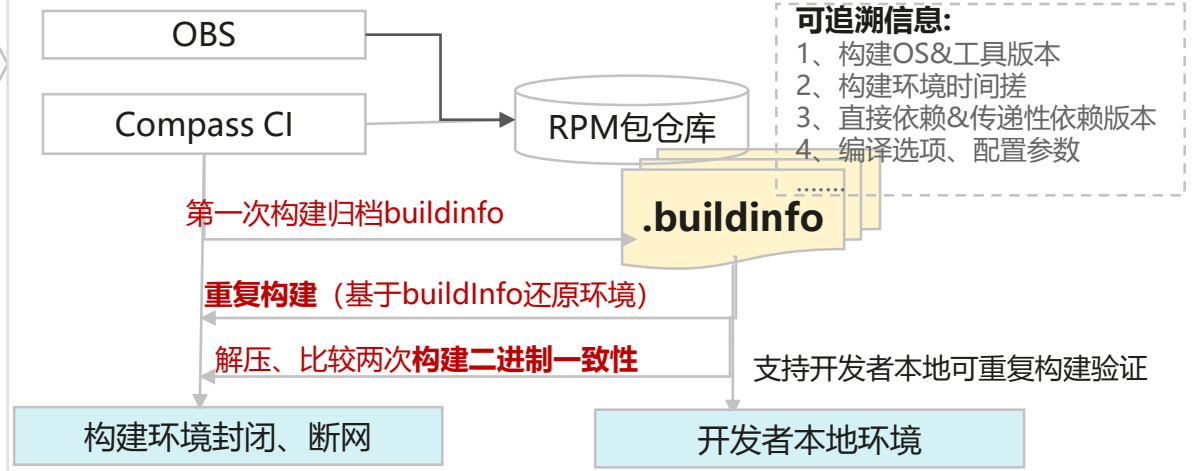
措施

成立社区可重复构建sig组

- **加入可重复构建官方组织** reproducible-builds.org
- 优化发布打包policy规范
- 可重复构建工具链方案
- sig组间的协同运作机制

建设可追溯、可重复社区工具链

可视化运营度量、趋势 | Issue自动推送、进展跟踪 | 问题自动定位、辅助修复



推动上游社区修复

- 建立上游社区PR审核、修复渠道
- 提供传递性依赖影响的修复方案
- 上游社区修复、自动验证工具链

计划

2022.Q1
- ✓ 成立可重复构建sig组、加入社区
- ✓ 工具链方案设计

2022.Q2
- ✓ 工具链试点、支持openEuler核心包扫描报告生成
- ✓ 在openEuler sig组运作推动上游社区修复问题

2022.Q3
- ✓ 工具链推广到openGauss/MindSpore/鸿蒙等
- ✓ 在Sig组运作推动上游社区修复问题

2022.Q4
- ✓ openEuler 实现1500+核心软件包达成95%
- ✓ openGauss/MindSpore达成95%

2022年可重复构建需求分解

Epic	目标	Story	责任Owner
开源社区可重复构建方案建设（公共能力）	2022年在各社区成立可重复构建专项运作、 并得到官方组织reproducible-builds.org认可	各社区技术委员会汇报、与开源社区&合作厂商达成共识、并制定验收标准	刘波、徐亮 、吴峰光
		整体可重复构建社区运作机制：在可重复构建社区标准章程	刘波、徐亮 、吴峰光
		整体工程技术方案：从编译、归档二进制和元数据、到二进制比较、到趋势看板等工具方案	刘波、徐亮 、吴峰光
		与官方组织reproducible-builds社区建立合作关系：梳理对官方组织的诉求、工具复用、与组织对我们的要求	徐亮
		可重复构建客户端工具开发：锁定时间搓、路径、环境、链接顺序、支持各构建系统集成	刘路
		可重复构建服务端网站参考： https://reproducible-builds.org/citests/： 软件列表页面、构建统计页面、每周&每月可重复率趋势图、问题状态页面、自动提交issue给个sig组	刘路
openEuler可重复构建（落地）	2022年核心包达成95%可重复 2023年全量包达成95%可重复 2024年98%可重复	社区可重复构建SIG组、定义可重构构建规范、各SIG组协同关系、建立推动上游社区修复二进制不一致问题的机制和渠道	刘波、吴峰光、徐亮
		对应未打成RPM包的各包仓库如jar包，tgz,whl,zip包传递性依赖进行整改成RPM包	吴峰光
		由于源码导致的二进制不一致问题、推动上游社区整改	吴峰光
		可重复构建客户端工具集成:到各构建系统如OBS,compassci	刘路
		构建元数据采集工具：自动生成buildInfo含构建环境、源码、依赖、时间、构建工程参数等， 基于归档构建元数据还原构建环境和构建工程可重复构建	刘路、吴峰光
		可重复构建环境封闭：断其它外部网如各社区maven,npm，pypi等仓库	曹志
		可重复构建二进制包仓库建设：存储每日构建软件包、和构建元数据buildInfo、支持随时可还原	曹志、刘路

OpenEuler

Topics引导--有请徐亮


OpenEuler

可重复构建洞察和TOP问题

开源社区资深贡献者

徐亮

目录

可重复构建概念和研究

- 概念定义

对于可重复的构建，给定相同的源代码、构建环境和构建指令，任何人均可重建出比特级完全相同的指定制品。

- 技术要素

1. 编译套件确定性代码生成
2. 压缩库确定性输出
3. 编译系统工具和集成
4. 重复构建验证工具和平台

- 现实意义：创建从代码到制品的可独立验证路径，结合已有的代码发布签名、软件仓库签名、安全启动等技术，使开源代码从生产到使用的全过程可追溯成为可能。

学术论文

- *Trusting Trust - Reflections on Trusting Trust* (1984) — Ken Thompson. (PDF)
- *Fully Countering Trusting Trust through Diverse Double-Compiling* (2005/2009) — David A. Wheeler (PDF, …)
- *Functional Package Management with Guix* (2013) — Ludovic Courtès. […]
- *Reproducible and User-Controlled Software Environments in HPC with Guix* (2015) — Ludovic Courtès, Ricardo Wurmus […]
- *in-toto: Providing farm-to-table guarantees for bits and bytes* (2019) — Santiago Torres-Arias, New York University; Hammad Afzali, New Jersey Institute of Technology; Trishank Karthik Kuppusamy, Datadog; Reza Curtmola, New Jersey Institute of Technology; Justin Cappos, New York University. (PDF)
- *Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks* (2020) — Marc Ohm, Henrik Plate, Arnold Sykosch, Michael Meier. (PDF)
- *Automated Localization for Unreproducible Builds* (2018) — Zhilei Ren, He Jiang, Jifeng Xuan, Zijiang Yang. (PDF)
- *Reproducible Containers* (2020) — Navarro Leija, Omar S. and Shiptoski, Kelly and Scott, Ryan G. and Wang, Baojun and Renner, Nicholas and Newton, Ryan R. and Devietti, Joseph. (…)
- *Towards detection of software supply chain attacks by forensic artifacts* — Marc Ohm, Arnold Sykosch, Michael Meier. (Link)

可重复构建社区的历史

可重复构建并非新生事物：

- 1992 年 GNU 工具链已有工具对此提供初步支持
- 2000 年 Debian 开发者提出了关于内嵌时间戳影响可重复性的技术讨论
- 2007 年 Debian 项目开始了全面支持可重复构建的思考
- 2013 年 Snowden 事件使社会对这个问题的关注度大幅提升；研究人员通过修改的 Xcode 实现了攻击，随后 XcodeGhost 被发现。
- 同年的Debian 开发者会议(Debconf13)过程中，可重复构建专题启动到八月完成首轮盲测5240个软件，可重复率仅24%

- *To*: winnie@der-winnie.de
- *Cc*: debian-devel@lists.debian.org
- *Subject*: Re: Building packages three times in a row
- *From*: Martin Uecker <muecker@gmx.de>
- *Date*: Sun, 23 Sep 2007 23:32:59 +0200
- *Message-id*: <[🔎] 1190583179.14192.11.camel@bluestein>
- *In-reply-to*: <[🔎] 200709182336.12740.winnie@der-winnie.de>

```
Patrick Winnertz wrote:
> Am Dienstag, 18. September 2007 21:12:44 schrieb Julien Cristau:
> > > Hmmhh, what do you do about programs etc that encode the build-time in
> > > the binary? I mean they obviously will change between builds?
> >
> > Hopefully they don't encode the build-time in the file list?
> We checked not for files which differ, but only for files which are missing
> in the first package. or which are missing in the second package.
>

I think it would be really cool if the Debian policy required
that packages could be rebuild bit-identical from source.
At the moment, it is impossible to independly verify the
integricity of binary packages.


Greetings,
Martin
```

进一步推动：

- 2014年 binutils 代码确定性生成补丁合入，二轮盲测6887个软件，达成率提高至 67%；这一年的Debian开发者会议(Debconf14)上敲定了可重复构建元数据格式，完成了基础集成工具的功能打样
- 2015年 FOSDEM15和Debconf15上，相关工具支持纳入 Debian 官方基础设施，并作为新版 Release Goal，此时全仓库达成率已推进至 83%
- 这一年还首次实现了 Firefox 的可重复构建；Coreboot, OpenWrt, NetBSD, FreeBSD, Archlinux 相继加入计划
- 2016-2017年，可重复构建纳入Debian社区官方打包规范 (4.1.0版本)，同时全仓库达成率推进至90%
- 2022年发布的Debian 11中，全仓库达成率94.8%(amd64, 31500 packages)

可重复构建社区当前概况

Reproducible builds 项目启动于 2013 年的 Debian 年度开发者会议，后作为独立社区托管于 Software Freedom Conservancy 基金会。

主要支持社区

Arch Linux
(79.2% of 10712)

Coreboot
(100% of 162)

Debian
(94.8% of 31500)

FreeBSD
(n/a)

NetBSD
(98.3% of 60)

OpenWrt
(99.2% of 828)

合作社区

GNU Guix
(79.43% of 17663)

NixOS
(99.87% of 1572)

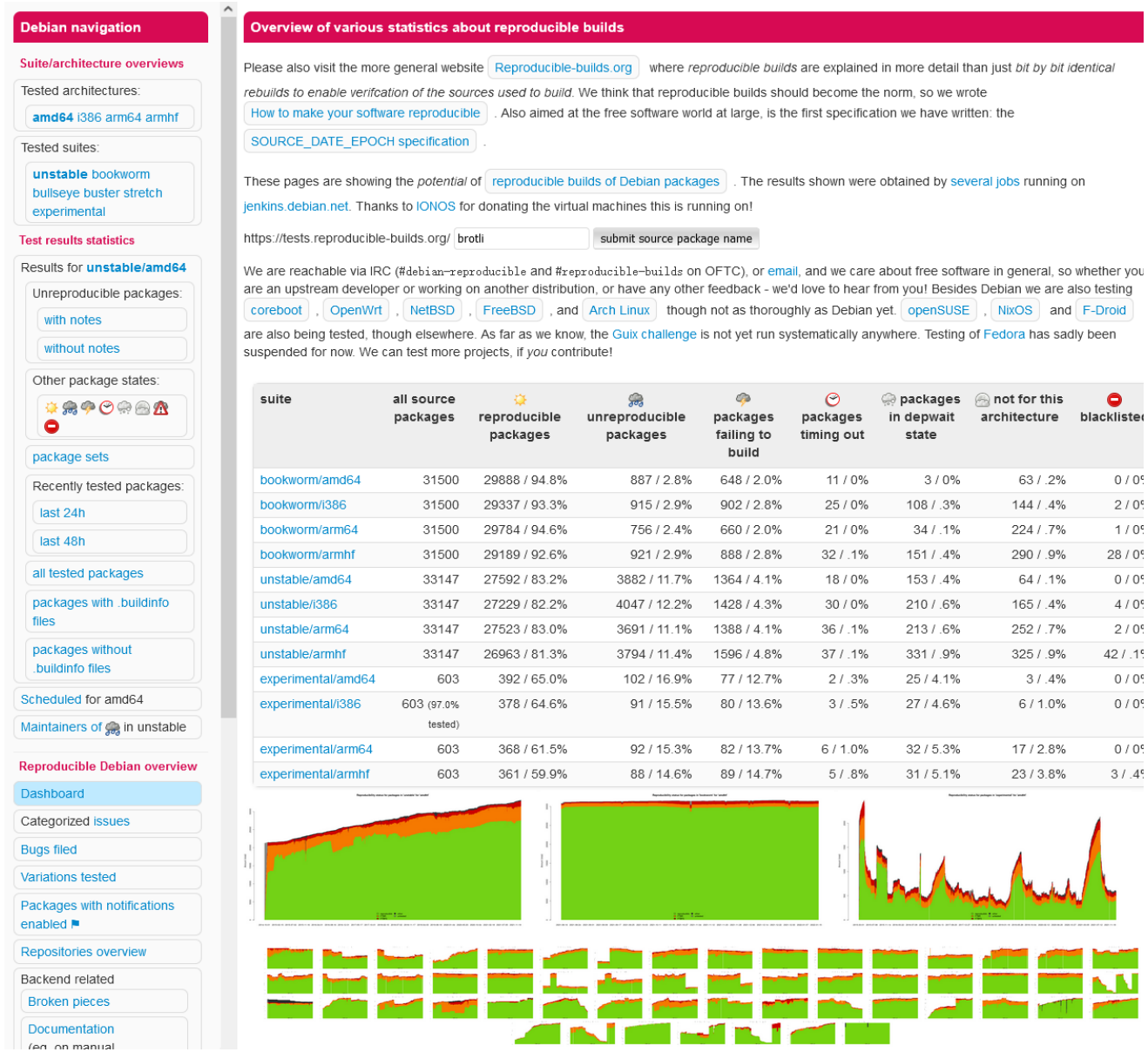
openSUSE
(91.86% of 14352)

Qubes OS
(n/a)

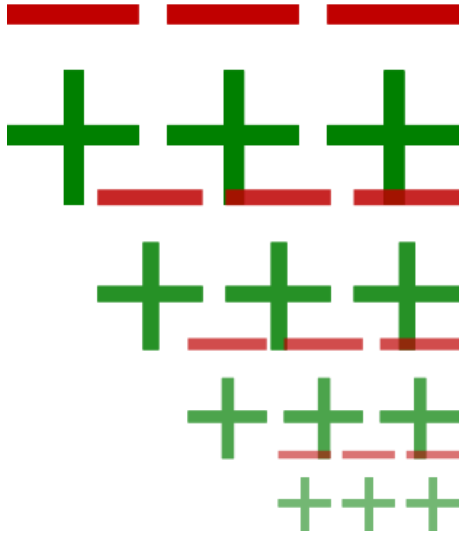
Yocto Project
(n/a)

社区现有工具和CI看板

名称	功能
diffoscope	差异比较工具
trydiffoscope	使用diffoscope的在线服务平台
disorderfs	非确定性文件系统注入工具
strip-nondeterminism	非确定性编译系统后处理工具
reprotest	可重复构建验证工具
rebuilderd	发行版仓库监控和验证工具
archlinux-repro	Archlinux验证后端
reproducible-build-maven-plugin	Apache Maven 构建系统插件
sbt-reproducible-builds	Scala sbt 构建系统插件



社区主要工具介绍 —— diffoscope



diffoscope 是一款用于深入比较文件、文件夹以及压缩包区别的工具。

主要功能是对各种包文件进行递归解压，并对内容进行比较，可支持海量包格式文件并具备模糊匹配功能，能够输出文本或HTML 格斯。

在线试用 https://try.diffoscope.org

```
frank@mauritius:~/Downloads$ diffoscope fp-utils*
 |#################################################|  100%                          Time: 0:00:01
--- fp-utils-3.0.0_3.0.0+dfsg-11+deb9u1_amd64.deb
+++ fp-utils_3.0.0+dfsg-11+deb9u1_amd64.deb
├── file list
 @@ -1,3 +1,3 @@
  -rw-r--r--   0        0        0        4 2017-06-10 17:13:48.000000 debian-binary
 --rw-r--r--   0        0        0     4248 2017-06-10 17:13:48.000000 control.tar.gz
 --rw-r--r--   0        0        0  2296496 2017-06-10 17:13:48.000000 data.tar.xz
 +-rw-r--r--   0        0        0     1113 2017-06-10 17:13:48.000000 control.tar.gz
 +-rw-r--r--   0        0        0    42708 2017-06-10 17:13:48.000000 data.tar.xz
├── control.tar.gz
├── 1-content
├── file list
│   @@ -1,5 +1,3 @@
│    drwxr-xr-x   0 root         (0) root         (0)        0 2017-06-10 17:13:48.000000 ./
│    --rw-r--r--   0 root         (0) root         (0)     1616 2017-06-10 17:13:48.000000 ./control
│    --rw-r--r--   0 root         (0) root         (0)     5553 2017-06-10 17:13:48.000000 ./md5sums
│    --rwxr-xr-x   0 root         (0) root         (0)     4677 2017-06-10 17:13:48.000000 ./postinst
│    --rwxr-xr-x   0 root         (0) root         (0)      142 2017-06-10 17:13:48.000000 ./prerm
│    +-rw-r--r--   0 root         (0) root         (0)     1431 2017-06-10 17:13:48.000000 ./control
│    +-rw-r--r--   0 root         (0) root         (0)      351 2017-06-10 17:13:48.000000 ./md5sums
├── ./control
│   @@ -1,30 +1,28 @@
│    -Package: fp-utils-3.0.0
│    +Package: fp-utils
│     Source: fpc
│     Version: 3.0.0+dfsg-11+deb9u1
│     Architecture: amd64
│     Maintainer: Pascal Packaging Team <pkg-pascal-devel@lists.alioth.debian.org>
│    -Installed-Size: 23493
│    -Depends: fpc-source-3.0.0, libc6 (>= 2.2.5)
│    -Recommends: fp-compiler-3.0.0 (= 3.0.0+dfsg-11+deb9u1)
│    -Breaks: fp-compiler (<= 2.4.0-3), fp-units-gfx (<= 2.4.2-2), fp-units-gfx-2.4.2 (<= 2.4.2-2), fp-utils (<= 2.4.0-3), fpc (<= 3.0.0+df
│    -Replaces: fp-compiler (<= 2.4.0-3), fp-utils (<= 2.4.0-3), fpc (<= 3.0.0+dfsg-0)
│    -Provides: fp-utils
│    +Installed-Size: 120
│    +Depends: fp-utils-3.0.0 (= 3.0.0+dfsg-11+deb9u1)
│     Section: devel
│     Priority: optional
│    +Multi-Arch: same
│     Homepage: http://www.freepascal.org/
│    -Description: Free Pascal - utilities
│    +Description: Free Pascal - utilities dependency package
```

https://diffoscope.org/examples/https-everywhere-5.0.6_vs_5.0.7.html

/tmp/https-everywhere-5.0.6.xpi vs.
/tmp/https-everywhere-5.0.7.xpi

zipinfo {}

Offset 1, 9 lines modified		Offset 1, 9 lines modified	
1	Zip file size: 2022489 bytes, number of entries: 194	1	Zip file size: 2018233 bytes, number of entries: 194
2	?rw------- 2.0 unx 73759 b- defN 80-Jan-01 00:00 Changelog	2	?rw------- 2.0 unx 73867 b- defN 80-Jan-01 00:00 Changelog
3	?rw------- 2.0 unx 4348 b- defN 80-Jan-01 00:00 chrome.manifest	3	?rw------- 2.0 unx 4348 b- defN 80-Jan-01 00:00 chrome.manifest
4	?rw------- 2.0 unx 575 b- defN 80-Jan-01 00:00 chrome/content/about.js	4	?rw------- 2.0 unx 575 b- defN 80-Jan-01 00:00 chrome/content/about.js
5	?rw------- 2.0 unx 3421 b- defN 80-Jan-01 00:00 chrome/content/about.xul	5	?rw------- 2.0 unx 3421 b- defN 80-Jan-01 00:00 chrome/content/about.xul
6	?rw------- 2.0 unx 6924 b- defN 80-Jan-01 00:00 chrome/content/code/AndroidUI.jsm	6	?rw------- 2.0 unx 6924 b- defN 80-Jan-01 00:00 chrome/content/code/AndroidUI.jsm
7	?rw------- 2.0 unx 9846 b- defN 80-Jan-01 00:00 chrome/content/code/ApplicableList.js	7	?rw------- 2.0 unx 9846 b- defN 80-Jan-01 00:00 chrome/content/code/ApplicableList.js
8	?rw------- 2.0 unx 12496 b- defN 80-Jan-01 00:00 chrome/content/code/ChannelReplacement.js	8	?rw------- 2.0 unx 12496 b- defN 80-Jan-01 00:00 chrome/content/code/ChannelReplacement.js
9	?rw------- 2.0 unx 4198 b- defN 80-Jan-01 00:00 chrome/content/code/Cookie.js	9	?rw------- 2.0 unx 4198 b- defN 80-Jan-01 00:00 chrome/content/code/Cookie.js
Offset 187, 10 lines modified		Offset 187, 10 lines modified	
187	?rw------- 2.0 unx 364 b- defN 80-Jan-01 00:00 chrome/skin/loop.png	187	?rw------- 2.0 unx 364 b- defN 80-Jan-01 00:00 chrome/skin/loop.png
188	?rw------- 2.0 unx 34968 b- defN 80-Jan-01 00:00 chrome/skin/ssl-observatory-messy.jpg	188	?rw------- 2.0 unx 34968 b- defN 80-Jan-01 00:00 chrome/skin/ssl-observatory-messy.jpg
189	?rw------- 2.0 unx 344 b- defN 80-Jan-01 00:00 chrome/skin/tick-moot.png	189	?rw------- 2.0 unx 344 b- defN 80-Jan-01 00:00 chrome/skin/tick-moot.png
190	?rw------- 2.0 unx 348 b- defN 80-Jan-01 00:00 chrome/skin/tick.png	190	?rw------- 2.0 unx 348 b- defN 80-Jan-01 00:00 chrome/skin/tick.png
191	?rw------- 2.0 unx 34424 b- defN 80-Jan-01 00:00 components/https-everywhere.js	191	?rw------- 2.0 unx 34424 b- defN 80-Jan-01 00:00 components/https-everywhere.js
192	?rw------- 2.0 unx 34854 b- defN 80-Jan-01 00:00 components/ssl-observatory.js	192	?rw------- 2.0 unx 34854 b- defN 80-Jan-01 00:00 components/ssl-observatory.js
193	?rw------- 2.0 unx 2509 b- defN 80-Jan-01 00:00 defaults/preferences/preferences.js	193	?rw------- 2.0 unx 2509 b- defN 80-Jan-01 00:00 defaults/preferences/preferences.js
194	?rw------- 2.0 unx 6393856 b- defN 80-Jan-01 00:00 defaults/rulesets.sqlite	194	?rw------- 2.0 unx 6366208 b- defN 80-Jan-01 00:00 defaults/rulesets.sqlite
195	?rw------- 2.0 unx 3157 b- defN 80-Jan-01 00:00 install.rdf	195	?rw------- 2.0 unx 3157 b- defN 80-Jan-01 00:00 install.rdf
196	194 files, 7422478 bytes uncompressed, 1993203 bytes compressed: 73.2%	196	194 files, 7394938 bytes uncompressed, 1988947 bytes compressed: 73.1%

社区主要工具介绍 —— disorderfs, strip-nondeterminism, reprotest

- disorderfs 是一个采用FUSE实现的，文件系统非确定性注入工具。例如，将文件系统对目录和文件列表的返回顺序进行随机化。
  查看代码：https://salsa.debian.org/reproducible-builds/disorderfs

- strip-nondeterminism 是一个用于去除如时间戳、文件系统顺序依赖等问题的 Perl 工具库，并作为 Debian 主力打包框架 debhelper 中的默认插件内置其中。
  查看代码：https://salsa.debian.org/reproducible-builds/strip-nondeterminism

- reprotest 是 Debian 项目使用的可重复构建验证工具框架，主要功能是在不同的环境中对代码进行编译，并使用 diffoscope、diff 等工具对产生的二进制文件进行对比、生成报告。
  查看代码：https://salsa.debian.org/reproducible-builds/reprotest

社区主要工具介绍 —— disorderfs, strip-nondeterminism, reprotest

- disorderfs 是一个采用FUSE实现的，文件系统非确定性注入工具。例如，将文件系统对目录和文件列表的返回顺序进行随机化。
  查看代码：https://salsa.debian.org/reproducible-builds/disorderfs

- strip-nondeterminism 是一个用于去除如时间戳、文件系统顺序依赖等问题的 Perl 工具库，并作为 Debian 主力打包框架 debhelper 中的默认插件内置其中。
  查看代码：https://salsa.debian.org/reproducible-builds/strip-nondeterminism

- reprotest 是 Debian 项目使用的可重复构建验证工具框架，主要功能是在不同的环境中对代码进行编译，并使用 diffoscope、diff 等工具对产生的二进制文件进行对比、生成报告。
  查看代码：https://salsa.debian.org/reproducible-builds/reprotest

影响可重复构建性的TOP问题

1. 确定性的软件构建过程
2. 定义标准化的构建环境
3. 构建信息的分发和重现
4. 制定对比协议

TOP 问题之一 —— 软件构建过程

问题	解释
构建系统自身设计	稳定的输入输出，尽量不从环境中获得输入
SOURCE_DATE_EPOCH	https://reproducible-builds.org/specs/source-date-epoch
不稳定的输入可能消失	例如构建过程中从别处下载的代码或工具
稳定的输入顺序	例如需要使用目录内所有文件的列表，不能依赖于文件系统的返回（不同文件系统实现的返回顺序不同，同一实现返回的顺序不稳定）
代码变量初始化	部分编程语言默认不对变量（或不保证）初始化，需提前赋值初始化
内嵌版本信息	对于软件内嵌的版本信息，应确保输入稳定，不使用日期、构建次数自增等方法
时间戳	时间戳不应从构建环境中获取，如果必须有，则应与源码产生的时间关联

TOP 问题之一 —— 软件构建过程

问题	解释
时区	使用统一的时区信息而非构建环境默认的
语言环境	使用统一的语言环境信息而非构建环境默认的
压缩包元数据	对于tar等工具，指定排序顺序、SOURCE_DATE_EPOCH、用户和组等；或使用工具后处理去除非确定性部分
稳定的输出顺序	部分数据类型不保证输出顺序，可设置如PYTHONHASHSEED等使其可重现
随机数	应当避免，确需使用应记录随机种子
构建路径	避免记录路径信息，如产生的 detached debug symbols 文件
系统镜像文件	云镜像、安装ISO等
JVM	尚在演进中，参考 https://github.com/jvm-repo-rebuild/reproducible-central#readme

TOP 问题之二 ——构建环境

构建环境的内容：

1. 指定操作系统环境
2. 编译环境的硬件架构
3. 指定编译发生的文件系统路径
4. 执行构建任务时采用的系统用户
5. 系统语言环境设置
6. 时区信息
7. 重要的环境变量（如SOURCE_DATE_EPOCH)

记录构建环境的要求和方法：

1. 环境自身所包含的组件可 100% 重复构建（组件自身可重复构建，组件版本信息和文件均记录归档，可在验证时获取）
2. 记录关于环境自身的元数据格式和工具，并能支持重现该环境

TOP 问题之三 —— 构建信息的分发和重现

1. 对外部代码，保存至本地并采用校验和验证
2. 或将外部代码完全提交至项目版本控制系统内（check-in everything)
3. 将满足要求的开发环境作为SDK提供给开发者
4. 定义标准的机读数据格式，记录软件的构建环境和构建信息：
   https://manpages.debian.org/unstable/dpkg-dev/deb-buildinfo.5.en.html

FIELDS

Format: *format-version* (required)
The value of this field declares the format version of the file. The syntax of the field value is a version number with a major and minor component. Backward incompatible changes to the format will bump the major version, and backward compatible changes (such as field additions) will bump the minor version. The current format version is **1.0**.

Source: *source-name* [(*source-version*)] (required)
The name of the source package. If the source version differs from the binary version, then the *source-name* will be followed by a *source-version* in parenthesis. This can happen when the build is for a binary-only non-maintainer upload.

Binary: *binary-package-list* (required in context)
This folded field is a space-separated list of binary packages built. If the build is source-only, then the field is omitted (since dpkg 1.20.0).

Architecture: *arch-list* (required)
This space-separated field lists the architectures of the files currently being built. Common architectures are **amd64**, **armel**, **i386**, etc. Note that the **all** value is meant for packages that are architecture independent. If the source for the package is also being built, the special entry **source** is also present. Architecture wildcards must never be present in the list.

Version: *version-string* (required)
Typically, this is the original package's version number in whatever form the program's author uses. It may also include a Debian revision number (for non-native packages). The exact format and

TOP 问题之四 —— 制定对比协议

1. 确定对比的内容，例如是否要求对于压缩包应解尽解后，对比解压后的具体内容；或直接对比交付用户的文件，如 .deb .rpm .iso 文件
2. 确定对比方法，如采用文件哈希值对比；同时给出对于确实无法在构建时复现信息的处理方法，如证书、签名等
3. 给出构建环境的重建方法，提供响应的工具和输入信息（如源代码、元数据），提供具体的对比工具

可重复构建SIG运作机制与任务

吴峰光

2022.4.12

可重复构建目标

- 一致性
  - 同样的输入，两次构建，得到完全一样的二进制输出
  - 允许环境差异：time, locale, kernel, etc.
  - 验证对象：构建工具、软件包spec及其上游软件
- 可信任
  - OSV构建平台的结果，第三方可利用开源工具本地复现
  - 验证对象：OSV

openEuler社区 开源协作

- 基于开源gitee repo进行开发
- 复用已有开源工具、平台
  - 工具：diffoscope
  - 平台：Compass CI, OBS

openEuler社区 双周SIG会议

- 讨论需求
- 研发方案
- 分析数据（统计量、典型情况）
- 推进解决问题

reproducible-builds.org 合作

- 工具复用
- 社区交流
- 创建对openEuler可重复构建的介绍与链接页面

-- 徐亮

对外界面

- 统计数据与图表页面（辅助管理决策）
- 问题清单、日志页面（辅助开发者fix）
- 构建、测试任务页面（平台集成）

- 月度社区报告
- 对单包维护者的问题报告 （问题在spec）
- 对上游开发者的问题报告 （问题在上游软件） （怎么判定？）

构建环境封闭

- 原则上所有RPM构建依赖也必须是RPM
- 依赖和输入必须来自RPM spec专用字段(Requires, BuildRequires, BuildPreReq, Sourcexxx, Patchxxx等)
- 依赖和输入必须有哈希验证

- 环境封闭用于避免如下最大的逃逸风险：
  禁止spec内%build等脚本动态下载额外依赖

可重复构建测试

- 批量、按需测试
- 日常PR门禁测试

- 新问题：禁止合入
- 老问题：定期提醒
- 头部问题：批量fix

buildinfo/SBOM

- 构建平台记录
- 第三方可复现 (本地运行开源构建工具)

-- 刘波

snapshots.openeuler.org

- 参照 snapshots.debian.org
- 内容不可变，长久保存
- 用于buildinfo引用，可重复构建所需依赖包稳定可用

-- 徐亮

- 存储空间
- Web服务

-- 基础设施/compass ci

交付节奏

- 2022: daily iso/repo
- 2023: version iso/repo

可重复构建工程技术方案

开源社区工具链工程师

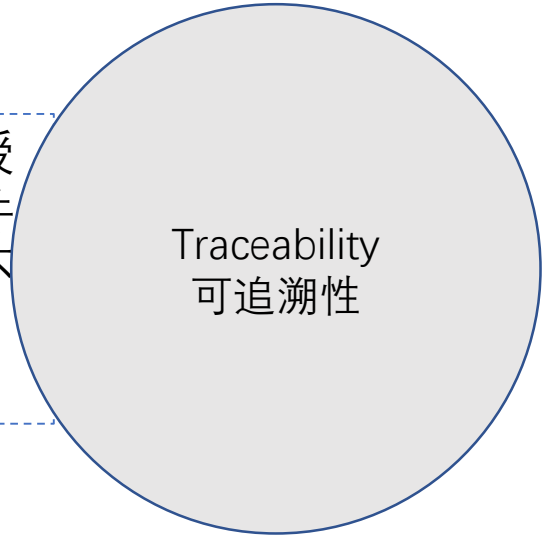
刘路

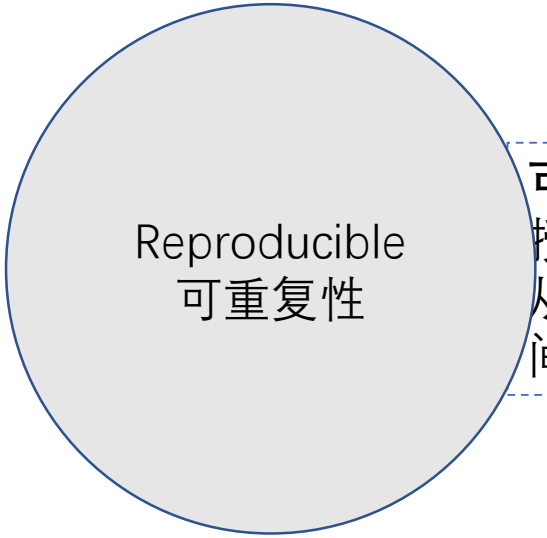
目录

可重复构建带来了什么收益

- 验证二进制未被植入后门，避免潜在安全风险；二进制质量保障
- 构建环境&构建工程能够被还原，依赖变化范围最小化、测试最小化、方便问题定位提高开发效率

可追溯性：任何人在获得授权的前提下，能够找到软件的任何变更历史和信息、不限于构建环境、构建工程、源代码、依赖信息等



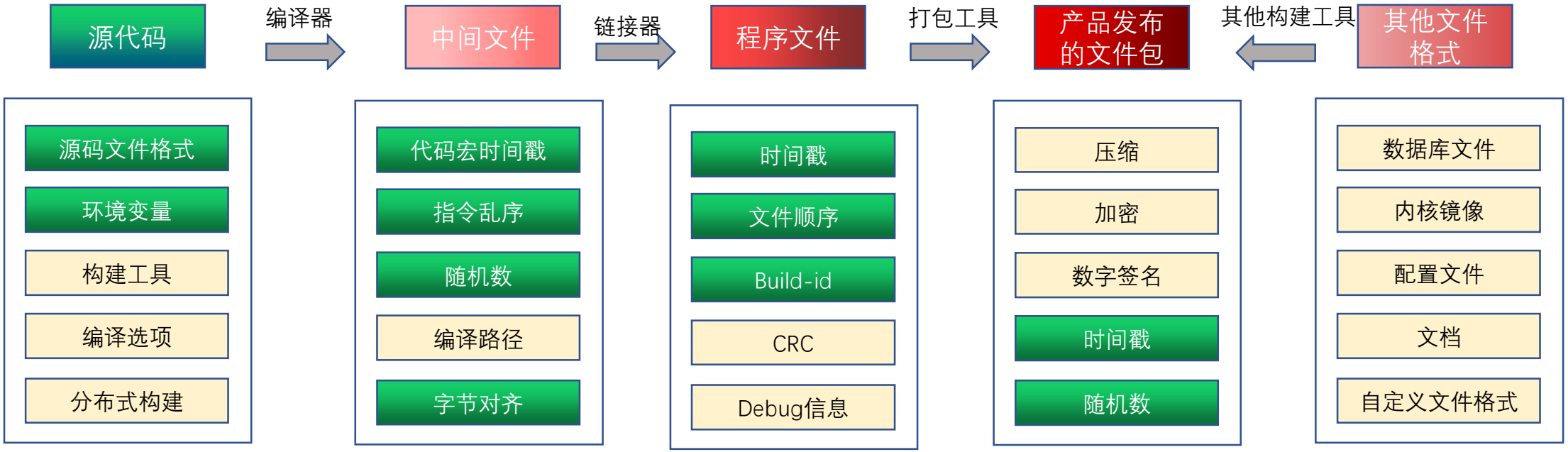


可重复性：任何人在获得授权的前提下，能够重现从过去到现在之间任意时间点的状态

不是通过手工归档构建信息实现重复编译，而是通过自动记录过程信息，并能够在任意时间点自动还原

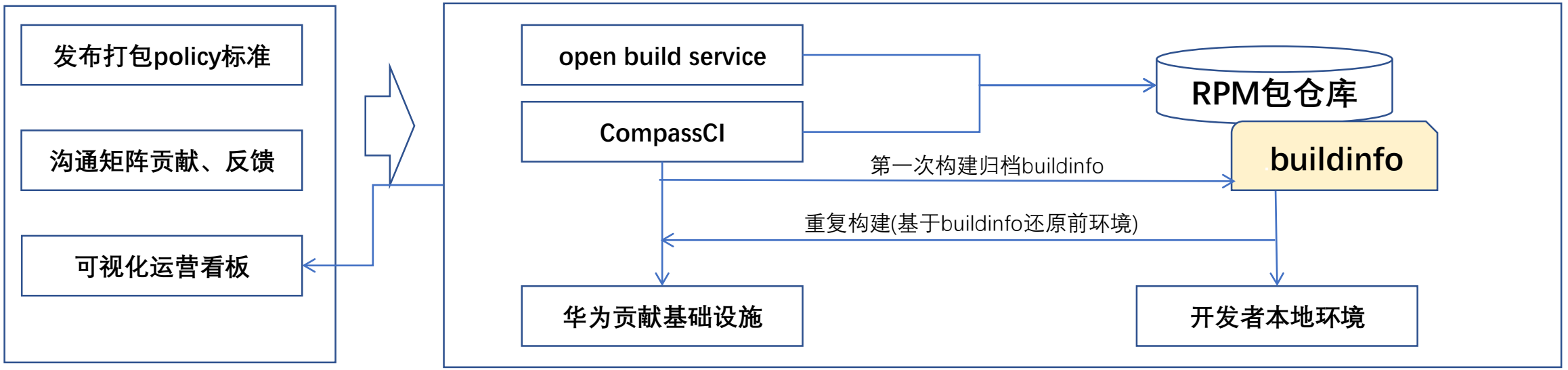
什么会影响可重复构建

- 影响因子：
- 源代码节点发生变化
- 构建工具、OS版本号、参数、编译路径发生变化
- 高阶语言包管理的语义化版本号导致依赖软件版本变化
- 时间戳、时区发生变化
- 随机数
- 文件乱序



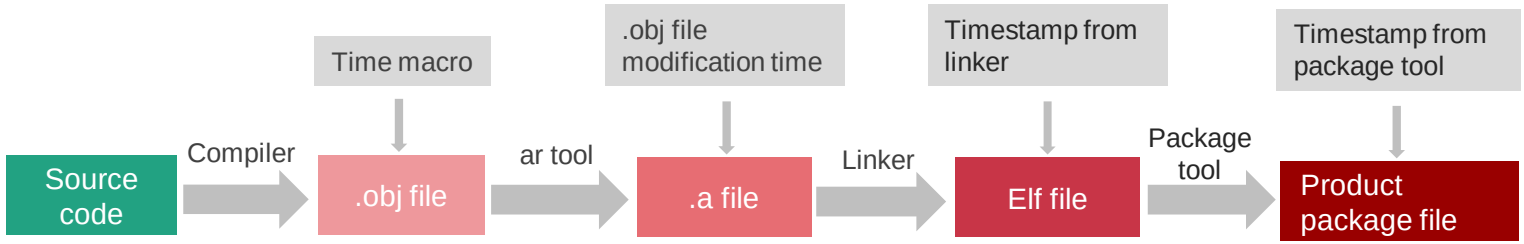
OpenEuler如何做重复构建

- **组织运作：**
- 组织加入：openEuler加入https://reproducible-builds.org/组织、共享共建服用已有社区能力
- 增加policy标准细化：加入影响可重复构建因素章节
- 可重复构建sig组：建立推动社区贡献 与Release&核心包Sig组联合运作、包括可重复问题可视、提交issue、修复、回合验证体系
- **工具链建设：**
- 测试流水线调度：基于commpassCI构建调度
- 执行工具链：buildinfo自动采集、引入开源二进制比较以及差异消除工具
- 可是运营体系：基于每个发行版&类型分组显示状态看板；邮件自动推送、issue自动提交、问题跟踪反馈



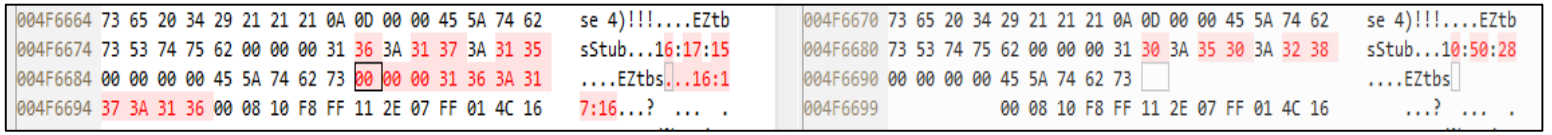
目录

为什么会有二进制差异

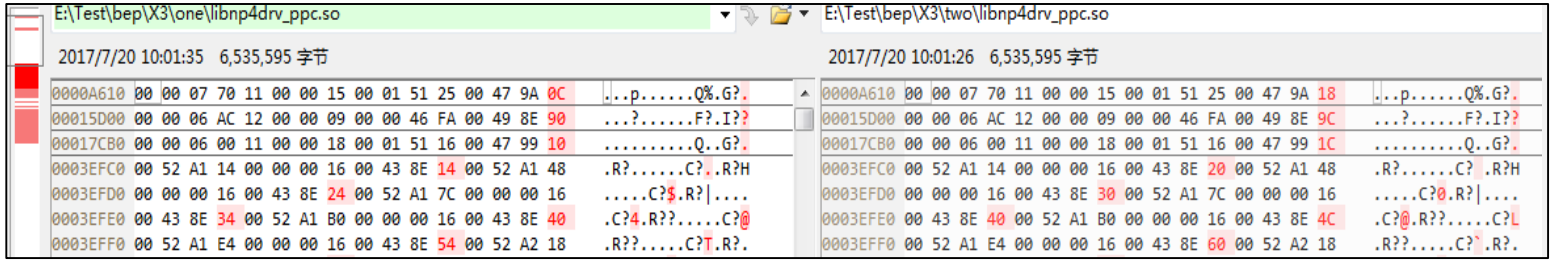


二进制差异案例（差异放大）

代码中的两个时间戳在编译时，偶发性的会合并成一个时间戳，如下图



而时间戳字符串的合并会造成ELF文件中data段后续的段地址发生偏移



挑战

- 在代码编译的各个阶段，都有可能会产生二进制差异。并且随着编译过程的累积，差异会逐渐放大，增加差异分析难度。
- 差异分析需要对编译器、构建过程、源码都十分熟悉，人工分析难度大，工作量高。

方案

- 相对于遗留二进制差异，并向客户解析差异原因，消除差异的投入要小很多
- 在构建其间消除二进制差异。

二进制差异分析

OpenEuler

类型	细类	差异原因	差异消除难度
时间戳	自研代码 开源代码	自研、开源代码中的_TIME_/_DATE_等时间宏导致二进制文件存在时间差异	通过整改代码实现消除时间戳差异。 通过编译选项可以消除时间宏差异，但会产生编译告警。
	商用工具 自研代码 开源工具	构建工具中调用_gettimeofday()、times()等函数生成时间信息并写入二进制	除构建工具本身提供的参数消除差异外，其他差异暂无消除方法。需要推动商业工具、开源工具消除差异，自研工具需要整改，不再生成时间差异。
随机差异	文档差异（office文档、TXT文档，chm帮助文档等）	txt文档中记录了二进制相关信息，包括Hash、二进制名称和版本等，如果二进制本身存在差异，生成的Hash值也会不一致	通过保证文档内容一致，其他由于文件格式导致的差异作为可解释差异。
	数字签名	数字签名带有时间戳，即使签名同样的文件，连续两次签名结果也并不一样	数字签名是为了保证安全校验，具有唯一性，建议作为可解释差异
	随机数 1）用时间戳作为随机数盐值 2）PID作为随机数 3）随机数生成器/dev/urandom 4）系统函数调用random()、rand() 5）buildID 6）CRC校验和 7）内存随机地址	随机数应用广泛，校验和、唯一的识别符、BuildID等，常用于唯一标识文件或二进制等	随机数差异只有部分差异可消除。如通过编译选项可以使gcc编译器不产生buildID差异。 但CRC校验和、内存随机地址差异、暂无特别有效的消除手段。如derbyDB这种数据库文件，会将当前构建服务器剩余内存写入二进制文件中。
	编译器导致的差异 1）汇编指令乱序 2）编译路径	编译器在编译源代码时，为了提升性能，会进行编译优化，会随机出现汇编指令乱序的差异。而且部分编译器会将编译源代码路径写入二进制文件中导致差异。	编译器编译优化导致的汇编指令乱序差异，当前可以去掉优化，或使用内存屏障技术消除差异。但是该方案当前只在mips和arm系统验证。编译路径差异暂无消除方案。
	文件乱序	多线程执行的任务会导致文件排序混乱	在使用构建工具执行构建前，对文件进行排序，可以消除排序差异。
	压缩算法	压缩工具会将二进制文件压缩，生成的二进制文件存在随机差异	部分压缩工具已提供了参数，用于生成没有差异的二进制文件。
	文件系统差异	文件系统的二进制文件，会存在多种类型的差异，如包含一些文件在磁盘位置的索引差异，压缩差异等。	文件系统二进制差异消除难度是最高的，仍然有很多无法消除的差异。

目录

链接顺序

环境会导致文件排序有差异

```
linux-v07s:/opt/BinCompare/ipbir_ipv6basic # export LANG=POSIX
linux-v07s:/opt/BinCompare/ipbir_ipv6basic # ls -al
total 18380
drwxr-xr-x 2 root root    4096 Oct 28 11:08 .
drwxr-xr-x 4 root root    4096 Oct 28 11:07 ..
-rw-r--r-- 1 root root   65704 Mar 29  2016 iaddr.o
-rw-r--r-- 1 root root   44656 Mar 29  2016 iaddr_fac.o
-rw-r--r-- 1 root root  880120 Mar 29  2016 icmp6.o
-rw-r--r-- 1 root root  720992 Mar 29  2016 icmp6_init.o
-rw-r--r-- 1 root root   37736 Mar 29  2016 icmp6_var.o
-rw-r--r-- 1 root root  707928 Mar 29  2016 iicmp6.o
-rw-r--r-- 1 root root  702712 Mar 29  2016 iicmp6_fac.o
-rw-r--r-- 1 root root   81648 Mar 29  2016 ind.o
-rw-r--r-- 1 root root   50160 Mar 29  2016 ind_fac.o
-rw-r--r-- 1 root root   34816 Mar 29  2016 ipmtu.o
-rw-r--r-- 1 root root  690960 Mar 29  2016 ipmtu_fac.o
-rw-r--r-- 1 root root  790440 Mar 29  2016 nd.o
```

```
linux-v07s:/opt/BinCompare/ipbir_ipv6basic # export LANG=en_US.UTF-8
linux-v07s:/opt/BinCompare/ipbir_ipv6basic # ls -al
total 18380
drwxr-xr-x 2 root root    4096 Oct 28 11:08 .
drwxr-xr-x 4 root root    4096 Oct 28 11:07 ..
-rw-r--r-- 1 root root   44656 Mar 29  2016 iaddr_fac.o
-rw-r--r-- 1 root root   65704 Mar 29  2016 iaddr.o
-rw-r--r-- 1 root root  720992 Mar 29  2016 icmp6_init.o
-rw-r--r-- 1 root root  880120 Mar 29  2016 icmp6.o
-rw-r--r-- 1 root root   37736 Mar 29  2016 icmp6_var.o
-rw-r--r-- 1 root root  702712 Mar 29  2016 iicmp6_fac.o
-rw-r--r-- 1 root root  707928 Mar 29  2016 iicmp6.o
-rw-r--r-- 1 root root   50160 Mar 29  2016 ind_fac.o
-rw-r--r-- 1 root root   81648 Mar 29  2016 ind.o
-rw-r--r-- 1 root root  690960 Mar 29  2016 ipmtu_fac.o
-rw-r--r-- 1 root root   34816 Mar 29  2016 ipmtu.o
```

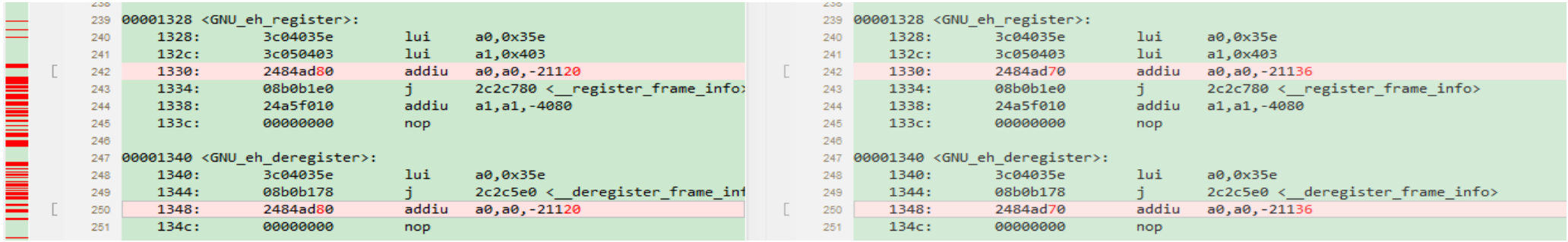
```
obj-y    := find *.o

EXTRA_LDFLAGS := -x

$(OS_OBJS): fpsp.h
```

```
obj-y    := bindec.o binstr.o decbin.o do_func.o gen_except.o get_op.o \
            kernel_ex.o res_func.o round.o sacos.o sasin.o satan.o satanh.o \
            scosh.o setox.o sgetem.o sint.o slog2.o slogn.o \
            smovecr.o srem_mod.o scale.o \
            ssin.o ssinh.o stan.o stanh.o sto_res.o stwotox.o tbldo.o util.o \
            x_bsun.o x_fline.o x_operr.o x_ovfl.o x_snan.o x_store.o \
            x_unfl.o x_unimp.o x_unsupp.o bugfix.o skeleton.o

EXTRA_LDFLAGS := -x

$(OS_OBJS): fpsp.h
```

解决方案：确保构建环境的环境变量信息在构建时保持固定。

时间戳合并

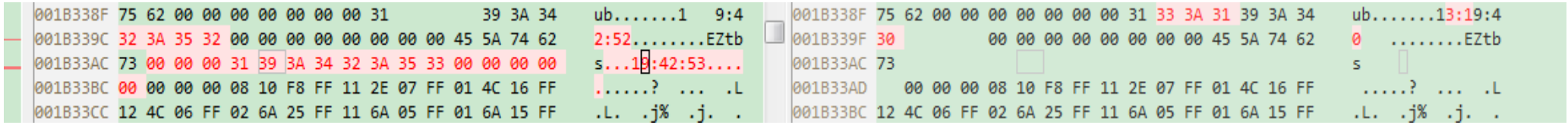
- 同一套代码编译生成的程序，代码段存在地址偏移的差异。



```
00001328 <GNU_eh_register>:
    1328:       3c04035e        lui     a0,0x35e
    132c:       3c050403        lui     a1,0x403
    1330:       2484ad80        addiu   a0,a0,-21120
    1334:       08b0b1e0        j       2c2c780 <__register_frame_info>
    1338:       24a5f010        addiu   a1,a1,-4080
    133c:       00000000        nop

00001340 <GNU_eh_deregister>:
    1340:       3c04035e        lui     a0,0x35e
    1344:       08b0b178        j       2c2c5e0 <__deregister_frame_info>
    1348:       2484ad80        addiu   a0,a0,-21120
    134c:       00000000        nop
```

```
00001328 <GNU_eh_register>:
    1328:       3c04035e        lui     a0,0x35e
    132c:       3c050403        lui     a1,0x403
    1330:       2484ad70        addiu   a0,a0,-21136
    1334:       08b0b1e0        j       2c2c780 <__register_frame_info>
    1338:       24a5f010        addiu   a1,a1,-4080
    133c:       00000000        nop

00001340 <GNU_eh_deregister>:
    1340:       3c04035e        lui     a0,0x35e
    1344:       08b0b178        j       2c2c5e0 <__deregister_frame_info>
    1348:       2484ad70        addiu   a0,a0,-21136
    134c:       00000000        nop
```

- 对比数据段，发现左侧文件有两个时间戳，右侧文件中只有一个时间戳。导致两个文件的数据段大小不同。



```
001B338F 75 62 00 00 00 00 00 00 00 31          39 3A 34    ub.......1   9:4
001B339C 32 3A 35 32 00 00 00 00 00 00 00 00 45 5A 74 62    2:52........EZtb
001B33AC 73 00 00 00 31 39 3A 34 32 3A 35 33 00 00 00 00    s...19:42:53....
001B33BC 00 00 00 00 08 10 F8 FF 11 2E 07 FF 01 4C 16 FF    ......?  ...  .L
001B33CC 12 4C 06 FF 02 6A 25 FF 11 6A 05 FF 01 6A 15 FF    .L.  .j%  .j.  .
```

```
001B338F 75 62 00 00 00 00 00 00 00 31 33 3A 31 39 3A 34    ub.......13:19:4
001B339F 30          00 00 00 00 00 00 00 00 45 5A 74 62    0   ........EZtb
001B33AC 73                                                 s
001B33AD    00 00 00 08 10 F8 FF 11 2E 07 FF 01 4C 16 FF     .....?  ...  .L
001B33BC 12 4C 06 FF 02 6A 25 FF 11 6A 05 FF 01 6A 15 FF    .L.  .j%  .j.  .
```

- 代码中有两个__TIME__宏，而左侧文件中时间戳展开成两个值;右侧文件中，两个宏展开成同一个时间戳。
- 数据段中的差异，会导致整个程序中函数地址、字符串地址等信息都会产生地址偏移。从而导致代码段也产生差异。

方案：1. 优化代码中时间戳宏。　2. 使用差异消除工具来设置时间戳。

字节对齐

- 二进制差异现象：

```
027C49EF 00 3B E3 00 00 40 82 00 0C 38 60 FF FF 48 00 00    .;..@..8`..H..
027C49FF 5C 80 63 00 28 38 80 FF FF 48 00 01 55 38 7F 00    \€c.(8€..H..U8▯.
027C4A0F 00 48 00 01 9D 38 7F 00 00 48 00 01 A5 80 7F 00    .H...▯..H.. ▯.
027C4A1F 28 48 00 00 BD 39 20 00 00 91 3F 00 28 38 7F 00    (H... ....(8▯.
027C4A2F 00 48 00 00 BD 3C 60 04 5A A0 63 12 48 3C 80 04    .H...`.Z燊.H<€.
027C4A3F 5A A0 84 12 4C 38 A0 00 01 38 DE 00 00 38 E0 00    Z燿.L8..8..8.
027C4A4F 00 48 00 00 DD 38 60 00 00 80 01 00 14 83 C1 00    .H...`..€...瓶.
027C4A5F 08 7C 08 03 A6 83 E1 00 0C 38 21 00 10 4E 80 00    .|..  ..8!..N€.
027C4A6F 20 25 73 3A 30 00 00 00 00 48 00 01 54 3D 80 01     %s:0....H..T=€.
027C4A7F 2D 39 8C D5 E4 7D 89 03 A6 4E 80 04 20 3D 80 01    -9屭鋧.  €. =€.
027C4A8F 34 39 8C 51 08 7D 89 03 A6 4E 80 04 20 3D 80 01    49陈.}.  €. =€.
027C4A9F 36 39 8C C9 30 7D 89 03 A6 4E 80 04 20 3D 80 01    69妮0}.  €. =€.
027C4AAF 3A 39 8C 97 30 7D 89 03 A6 4E 80 04 20 3D 80 01    :9窿0}.  €. =€.
027C4ABF 39 39 8C 9C 74 7D 89 03 A6 4E 80 04 20 3D 80 01    99寻t}.  €. =€.
```

```
027C49EF 00 3B E3 00 00 40 82 00 0C 38 60 FF FF 48 00 00    .;..@..8`..H..
027C49FF 5C 80 63 00 28 38 80 FF FF 48 00 01 55 38 7F 00    \€c.(8€..H..U8▯.
027C4A0F 00 48 00 01 9D 38 7F 00 00 48 00 01 A5 80 7F 00    .H...▯..H.. ▯.
027C4A1F 28 48 00 00 BD 39 20 00 00 91 3F 00 28 38 7F 00    (H... ....(8▯.
027C4A2F 00 48 00 00 BD 3C 60 04 5A A0 63 12 48 3C 80 04    .H...`.Z燊.H<€.
027C4A3F 5A A0 84 12 4C 38 A0 00 01 38 DE 00 00 38 E0 00    Z燿.L8..8..8.
027C4A4F 00 48 00 00 DD 38 60 00 00 80 01 00 14 83 C1 00    .H...`..€...瓶.
027C4A5F 08 7C 08 03 A6 83 E1 00 0C 38 21 00 10 4E 80 00    .|..  ..8!..N€.
027C4A6F 20 25 73 3A 30 00 00 27 7D 48 00 01 54 3D 80 01     %s:0..'}H..T=€.
027C4A7F 2D 39 8C D5 E4 7D 89 03 A6 4E 80 04 20 3D 80 01    -9屭鋧.  €. =€.
027C4A8F 34 39 8C 51 08 7D 89 03 A6 4E 80 04 20 3D 80 01    49陈.}.  €. =€.
027C4A9F 36 39 8C C9 30 7D 89 03 A6 4E 80 04 20 3D 80 01    69妮0}.  €. =€.
027C4AAF 3A 39 8C 97 30 7D 89 03 A6 4E 80 04 20 3D 80 01    :9窿0}.  €. =€.
027C4ABF 39 39 8C 9C 74 7D 89 03 A6 4E 80 04 20 3D 80 01    99寻t}.  €. =€.
```

- 第三方工具引入的差异:
  - device_t xbdRamDiskDevCreate (WINDER RIVER)

```
if (flag)
    {
    /*
     * Using partitions. Partition will be called <name>:0 and will
     * generate async insertion events. Therefore we need to wait
     * until the path gets finally instantiated.
     */
    sprintf(volName, "%s:0", name);
    fsmNameMap (volName, basePath);
    fsPathAddedEventSetup(&pathWait, basePath);
    }

erfEventRaise(xbdEventCategory, xrd->insertEventType, ERF_SYNC_PROC,
    (void *)device, NULL);
```

方案：修改第三方工具源码，或遗留差异.

时间戳

时间戳被广泛用于各种软件和系统中。

Linux OS information

```
linux-v07s:/opt/CTF/sccctf/crypto100 # uname -a
Linux linux-v07s 3.0.13-0.27-default #1 SMP Wed Feb 15 13:33:49 UTC 2012 (d73692
b) x86_64 x86_64 x86_64 GNU/Linux
```

CISCO Router version information

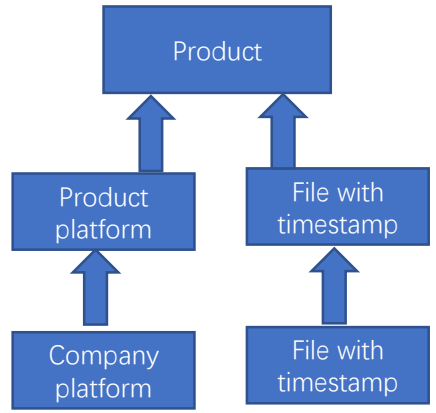
```
Cisco IOS Software, C3560E Software (C3560E-UNIVERSALK9-M), Version 15.2(1)E, RELEASE SOFTWARE (fc3)

Technical Support: http://www.cisco.com/techsupport

Copyright (c) 1986-2013 by Cisco Systems, Inc.

Compiled Tue 27-Aug-13 16:37 by prod_rel_team
```

IMAGE_FILE_HEADER in PE Head

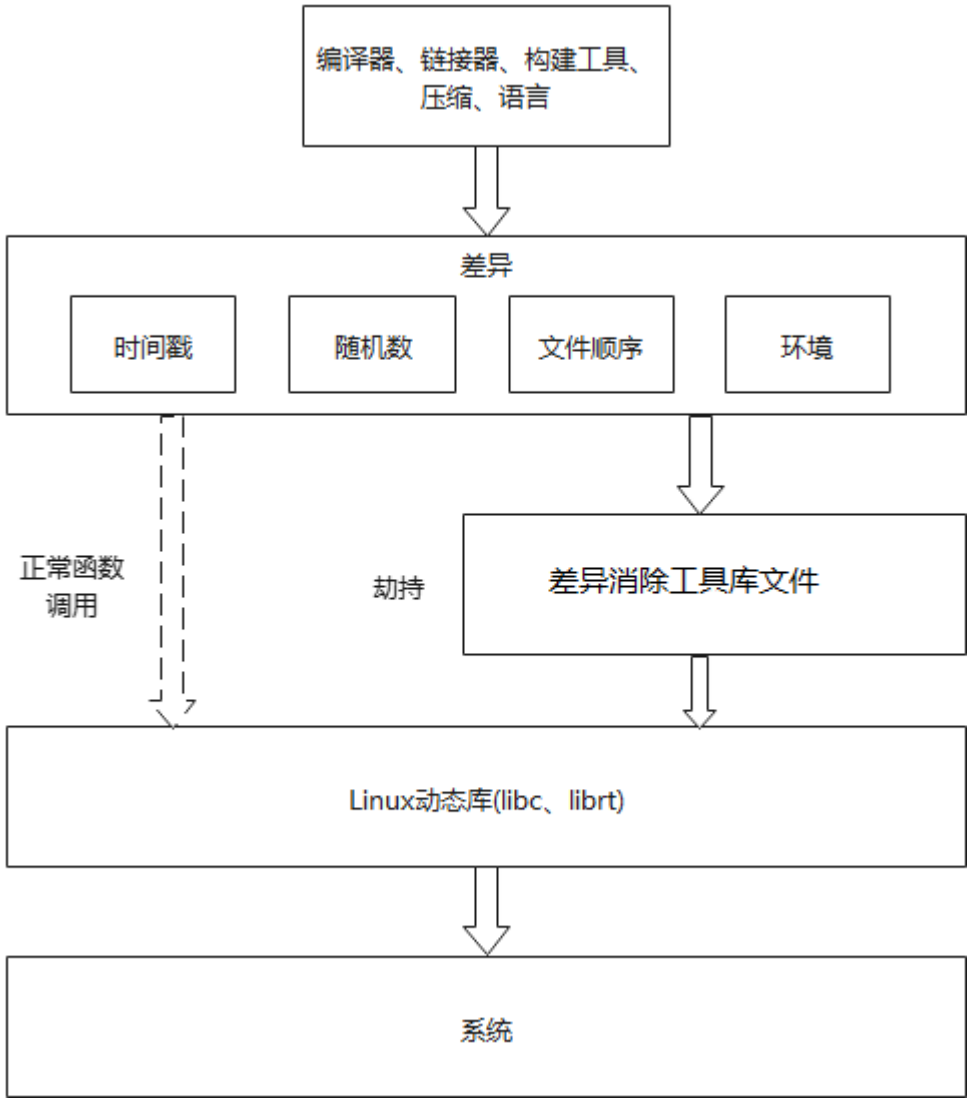
```
typedef struct _IMAGE_FILE_HEADER {
    WORD    Machine;
    WORD    NumberOfSections;
    DWORD   TimeDateStamp;
    DWORD   PointerToSymbolTable;
    DWORD   NumberOfSymbols;
    WORD    SizeOfOptionalHeader;
    WORD    Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

时间戳不只是以字符串的形式存在，还有数字、十六进制。

华为软件采用分层开发模型。公司级平台将带有时间戳的二进制文件传递到产品平台级。产品级平台还向产品交付带有时间戳的二进制文件。



差异消除工具原理



机制

- 差异消除工具,将环境变量LD_PRELOAD设置为在编译之前预加载`差异消除库文件`之后，该库将拦截系统时间的调用以确保编译状态的一致性。

过程

- 在编译代码之前，设置`差异消除库文件`保存的时间戳，将环境变量LD_PRELOAD设置为保存的时间戳，然后编译代码。
- 在重建阶段，将`差异消除库文件`设置成相同的时间戳并进行重建。

效果

- 产品源代码不需要修改，从而减少了产品线在代码纠正方面的投资。
- 商业和开放源代码工具所产生的差异也将得到消除，而无需进行任何代码修改。
- 时间戳的语义被最大程度地保存。
- 该工具支持32位和64位系统。
- 可以通过“ unset LD_PRELOAD”格式的命令禁用该工具。
- 在配置阶段应该做更多的工作。

谢谢