IoT theatre presents: The Tempest

Island of TEX (developers)

Abstract

2021 was a challenging year for the Island of TEX: roadmap changes, lack of resources, server limitations. Yet, resilience, persistence and a bit of good humour made the island even stronger, with new joiners, community support, bold plans and an even brighter future for the TEX ecosystem. And all just in time for celebrating 10 years of arara, our beloved bird!

1 Introduction

For those who do not know anything about us, the Island of TEX started as a pair of friends trying to improve the TEX ecosystem. Nowadays, the island acts as a friendly hub to community-based TEX-related projects, still mostly focused on the tooling side of things.

The last year has been challenging for us: roadmap changes, lack of resources, server limitations. Yet, resilience, persistence and, of course, a bit of good humour made the island even stronger, with new joiners, community support, bold plans and an even brighter future for the TEX ecosystem.

2 Docker images

The Island of TEX provides Docker images for easily reproducible builds as well as an official response to the need for continuous integration. Our images were among the first using vanilla TEX Live, providing the required tools for running software included in TEX Live — for example, Java Virtual Machine, Python, and so forth. Additionally, we provide TEX Live releases from 2014 on and let the user decide whether they want to pull all the documentation and source files into their CI configuration.

We officially publish our images at Docker Hub: `hub.docker.com/r/texlive/texlive`. Docker Hub is the world's largest repository of container images with an array of content sources including container community developers, open source projects and independent software vendors building and distributing their code in containers.

We feel honoured to maintain and provide one of the most complete and comprehensive TEX Live Docker images available in the TEX ecosystem. Several organisations now base their images on ours — for instance, DANTE e.V., the German-speaking TEX user group. Thank you very much for trusting our builds!

More recently, given certain characteristics of a typical TEX Live install, our shared CI runners could not meet the requirements to properly build and generate entries in our registry for historic and current images (e.g., failure due to timeout and disk space constraints). At first, we believed this was just a temporary issue (e.g., having a job assigned to a weak CI runner) and subsequent jobs would eventually have our images correctly built. Alas, we had no luck — failure rates were increasing at an alarming rate.

Vít Novotný reached out to us to discuss potential build improvements to our images and joined our Docker team. A new islander! His contributions were amazing — the build process soon became way more robust and reliable than ever. However, the blocking issue with our shared CI runners was still haunting us. We had to do something.

We agreed to reach out to the TEX community and ask for any spare computational resource that could host one of our CI runners. We then wrote an e-mail to the TEX Live mailing list on May 12 and hoped for the best.

In less than 30 minutes, we got three replies offering help! In that same day, the island already had a new dedicated CI runner. A couple of days later, two more runners were added to our pool. The response from the TEX community was fantastic.

Special thanks to Uwe Ziegenhagen, Erik Braun, DANTE, Marei Peischl, Paul Gessler, and the Institute of Mathematics of the Czech Academy of Sciences. We really appreciate it — our Docker images are saved because of you! Also, thanks to all who sent us messages of encouragement, from electronic mail to public support via our Matrix chat room.

Incidentally, Marei Peischl became an islander as well, joining our team of Docker experts. Thanks, Marei! Apparently, dragons are very good at building images!

Still, there is a long road to go. The TEX Live images are becoming more and more versatile but we still only feature full-fledged installations. So there are new frontiers waiting for us: splitting the images by scheme and improving layer-friendliness. Docker experts are invited to join our quest to solve these issues.

3 TEXdoc online

From all those projects using our TEX Live images, TEXdoc online is our most prominent contribution in the field — an online TEX and LATEX documentation lookup system based on CTAN's JSON API. Alas, TEXdoc online has not seen much activity in the last

doi.org/10.47397/tb/43-2/tb134island-tempest

year. It has a long way to go until it is not only a successor of the former `texdoc.net` but an even better replacement.

Some of our ideas include a source code lookup mode, based on a combination of texdoc, kpsewhich and file output, and runtime macro definition capabilities with a frontend for texdef. In general, the frontend should get a UI overhaul at some point in order to make it friendlier, more responsive, and more accessible.

We are also planning to include an optional analytics layer based on anonymous browsing and queries. But do not worry; at this point we are only interested in counting visits for starters — something which does not track anyone. Our generous sponsor of the world's most accessed TEXdoc online instance at `texdoc.org`, Stefan Kottwitz, would love to see this feature implemented as well. By the way, thanks for updating and maintaining `texdoc.org`, Stefan!

If anyone is interested in tackling one (or more) of the aforementioned features, definitely get in touch with us. All you need is some kind of programming background. Picking up Kotlin and contributing is something we can help you with. And if you have further ideas for improvement for this or any of our projects, our issue trackers are open 24/7. We believe that community feedback is key to building software the community actually finds useful.

4 CLI tooling

Apart from the web-based end, we should expect improvements to our CLI tooling in the near future as well. Take, for instance, our lovely Albatross, a tool for finding system fonts that provide a certain glyph (`ctan.org/pkg/albatross`).

Some of our ideas include report customisation, support for output formats (e.g., JSON or CSV), and caching. We also plan extending the glyph lookup based on specific files and directories, so any font in the filesystem — even ones not installed — could be properly inspected.

We've released a patched version of checkcites, our tool for checking missing or unused references, in order to address an outstanding issue due to a breaking BibLaTeX update. (`ctan.org/pkg/checkcites`)

This tool also has an interesting roadmap. Some of our ideas include a complete rewrite from Lua to Kotlin (work in progress) and moving to a modular approach, in which we have a proper BibTeX parser and a command line interface.

The ultimate goal for our BibTeX parser is to produce native code for all major operating systems alongside a Java Virtual Machine-compliant bytecode for major vendors, as well as a JavaScript backend.

This would be the perfect place to talk about our plans for arara as they share many goals. But before we do that, we want to celebrate with you. So let us go back in time for a bit.

5 Ten years of arara

Ten years ago, the very first version of arara, the cool TEX automation tool, was released. It was a humble flight for such a little bird. Little did we know, a delightful story about friendship, TEX, community and noisy birds was about to be written.

5.1 Version 1

There is a famous quote along the lines of

> *If at first you do not succeed,*
> *call it version 1.0.*

Version 1 of arara was also the first public release, dated April 4, 2012. Nothing much was there, besides the core concepts that still exist today: rules and directives.

Amusingly, the first version offered only a log output as an additional feature. There was no verbose mode. The log file was a gathering of streams (error and output) from the sequence of commands specified through directives. And that was it.

5.2 Version 2

The first version had a serious drawback: compilation feedback was not in real time and, consequently, no user input was allowed. For version 2, real time feedback was introduced when the tool was executed in verbose mode.

Two other features were included in this version: a flag to set an overall execution timeout, in milliseconds, as a means to prevent a potentially infinite execution, and a special variable in the rule context for handling cross-platform operations.

5.3 Version 3

So far, arara was only a tiny project with a very restricted user base. However, for version 3, a qualitative goal was reached: the tool became international, with localised messages in English, Brazilian Portuguese, German, Italian, Spanish, French, Turkish and Russian. Further, new features such as configuration file support and rule methods brought arara to new heights. As a direct consequence, the lines of code just about doubled from previous releases.

When the counter stopped at version 3, we decided it was time for arara to graduate and finally be released in TEX Live. Then things really changed in our lives. Given the worldwide coverage of TEX distributions, arara silently became part of the daily typographic tool belt of many users.

5.4 Version 4

Version 4 was definitely a quantum leap from previous releases. New features included a REPL workflow (i.e., rule evaluation on demand as opposed to prior to execution), an improved rule format, support for multiline directives, partial directive extraction mode, commands and triggers as abstraction layers, and an improved lookup strategy for configuration files.

5.5 Version 5

Version 5 featured a major rewrite from Java to Kotlin. We mainly worked on features from user feedback, especially directory support and the processing of multiple files.

We got hooked by the idea of aligning release schedules of arara with TEX Live releases enabling us to make (small) breaking changes more often. We had big plans and started to work on version 6 right after releasing version 5. As with checkcites, we walked the extra mile and arara went from a monolithic implementation to a modular one, plus an enhanced feature set and optimized workflow.

5.6 Version 6

Version 6 was split between an API, a core implementation, the engine and the CLI application to separate concerns. This was the first step in the direction of splitting out components that are bound to one platform. New features included specifying command line options to be passed to arara's session map, default preambles, expansion within directives, safe mode, and rule improvements (towards safety and optimization).

5.7 Version 7

For version 7, we wanted to take larger steps towards platform-independence. Some components had to be rewritten, some needed different interfaces for different platforms. It is still a heavy work in progress, but in the end, we hope to provide an even more multitalented tool.

Version 7 still targets JVM, but the tool is already being shaped towards platform-independence. New features include a new interface for the most common file operations (as a means to supersede Java's I/O API in the future), better error messages to indicate potential encoding problems, header mode enabled by default, and a brand new project specification.

6 The future

As previously stated, we are planning the stabilization of the current projects and implementation of new modular components, as well as improving support for our tools by producing native executables by means of Kotlin/Native, a technology for compiling Kotlin code to native binaries without the need of a JVM, languages like Rust and similar modern technology.

We also have challenges. For starters, hardware limitations for development and testing. The development happens on GNU/Linux machines. Incidental issues specific to Windows or macOS are handled through voluntary testing from users who sometimes do not have development expertise.

The lack of such systems in the development pool poses a problem and can hinder the long-term goals for better coverage and interoperability. See, for instance, the recent issues regarding Windows support in version 7 — we had to release three patches in quick succession to properly address these issues!

Also, the island has no documentation team, so we need to cover all fronts during development and release, not to mention the limited number of active developers and contributors.

Again, we kindly ask the TEX and software community for help, in any way possible. And, as always, thanks for the patience with us.

Also, a special thanks to our new members Jonathan Spratte, Marei Peischl and Vít Novotný! We are fortunate to have you on the team!

We have many plans and hope to realize as much as possible. The island is a vibrant environment for the development of TEX-related tools; we want to enhance the user experience, from newbie to expert, and promote use and diffusion of modern methodologies and technologies. If you are interested in helping us develop ideas or even implementing some code, visas for the island are free and no bureaucracy is involved, so feel free to reach out.

The Island of TEX is hosted at GitLab, whom we thank for providing us with a premium plan. It's highly appreciated.

If you are a TEX ecosystem tool author and want to join us, you and your projects are always welcome. If you want to become a tool author, or rewrite an existing tool, you are welcome as well!

◇ Island of TEX (developers)
https://gitlab.com/islandoftex