IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

RÉPUBLIQUE
FRANÇAISE
Liberté
Égalité
Fraternité

Reproducible LaTeX builds

Traceability, performance, sobriety. Pick 3.

Contents

Introduction

- A legacy of version control
- Current issues
- What's next?

A legacy of version control

The past:

- A dozen SVN repositories
- Heterogeneous workspaces
- Everything done “the SVN way”

A legacy of version control

The past:

- A dozen SVN repositories
- Heterogeneous workspaces
- Everything done “the SVN way”

The present:

- Deprecation of SVN within IMT Atlantique
- Migration to Git (with GitLab) in progress
- Everything still done “the SVN way”

Current issues

- Git is not made to store binary data
- Everything is compiled locally, it “works on my machine”™
- Poor dependency tracking, making it hard for newcomers

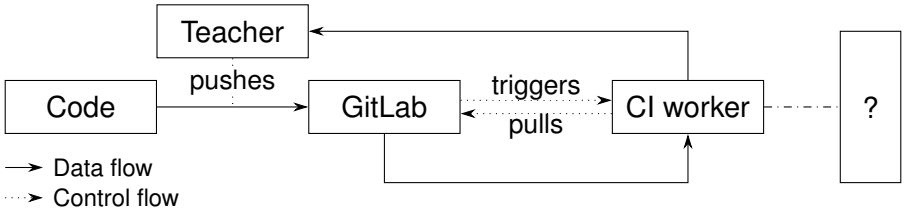
What's next?

- Avoid pushing PDF files to Git
- Use GitLab CI infrastructure
- **Reproduce** build artifacts

“Just use a CI”

- Continuous Integration 101
- The mystery box
- The need for proper tools

Continuous Integration 101



“Just use a CI”

The mystery box

```
make
```

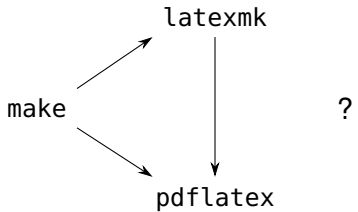
“Just use a CI”

The mystery box

- Rebuilds all the artifacts
- Can be really slow... `make`
- ... and very power hungry

The mystery box

- Rebuilds all the artifacts
- Can be really slow...
- ... and very power hungry



The need for proper tools

- Something is key in making the “mystery box” viable: **traceability**, more specifically, dependency tracking...

The need for proper tools

- Something is key in making the “mystery box” viable: **traceability**, more specifically, dependency tracking...
- ... and not disrupting existing workflows

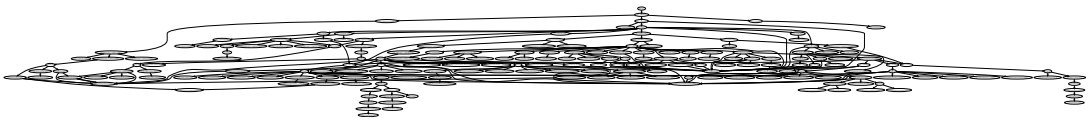
FLSdb

- Tracking dependencies
- Detecting change
- Architecture
- Demo

Tracking dependencies > The naïve way: C dependencies

- Source dependencies are explicitly declared in source files
- After a single run of the preprocessor, a dependency graph can be built
- The process is trivial

Tracking dependencies > The naïve way: C dependencies



- Source dependencies are not always explicitly declared
- Dependencies are fully known if the compilation is eventually idempotent, when idempotence is reached (some hypotheses can be relaxed)
- The dependency graph is flattened by LaTeX compilers, there is no concept of “compilation units”
- This graph is a free artifact of LaTeX compilations (`.fls` file)

Tracking dependencies > How LaTeX really works

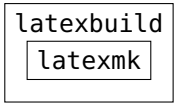
Detecting change

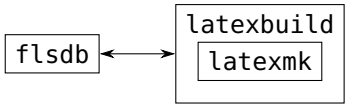
- Git workflows do not guarantee modification time monotonicity
- Computing file digests can be resource-intensive
- Again, we have no prior information on new source files

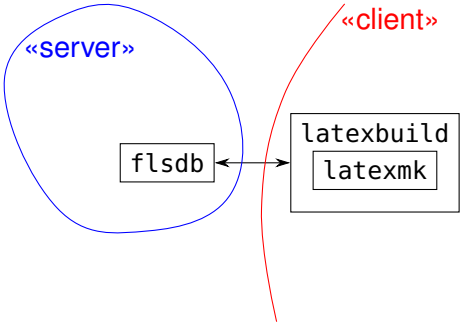
⇝ We need heuristics to optimize performance: we chose to compute the digest of the main source file and get the modification time of dependencies

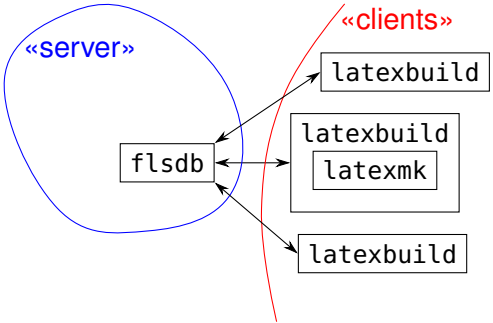
`latexmk`

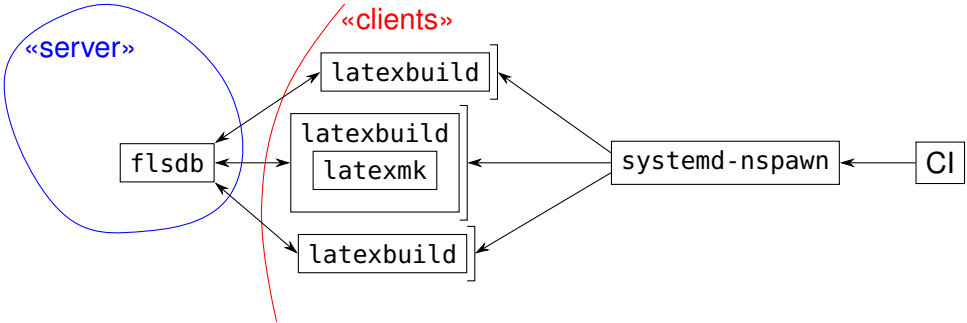
Architecture > CI instrumentation



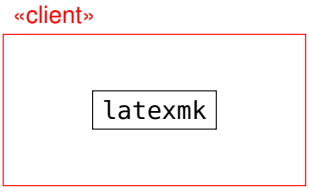

flsdb
latexbuild
latexmk


«server»
«client»
flsdb
latexbuild
latexmk

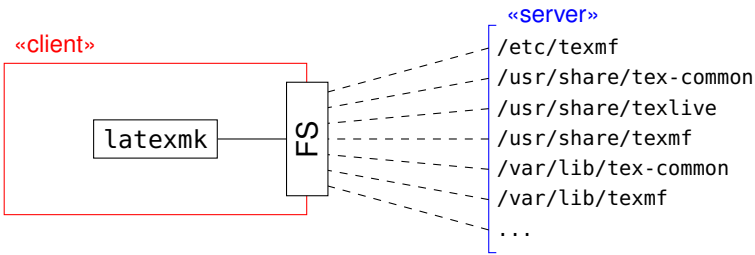

«server»
«clients»
flsdb
latexbuild
latexbuild
latexmk
latexbuild


«server»
«clients»
latexbuild
latexbuild
latexmk
flsdb
systemd-nspawn
CI
latexbuild

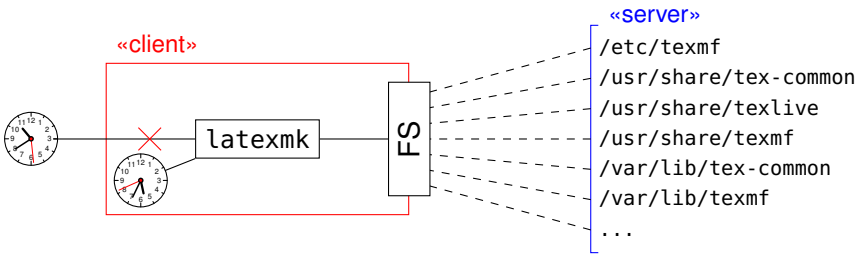
Architecture > LaTeX instrumentation



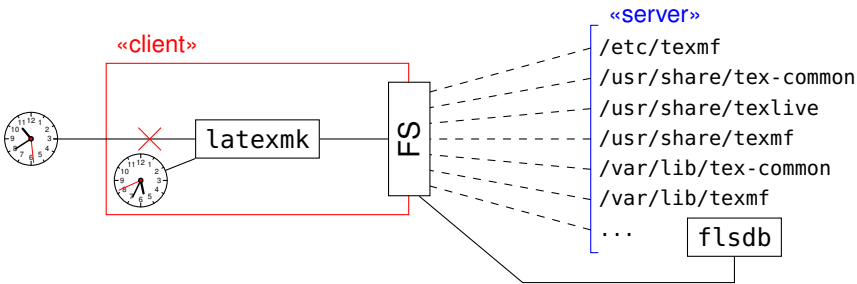
Architecture > LaTeX instrumentation



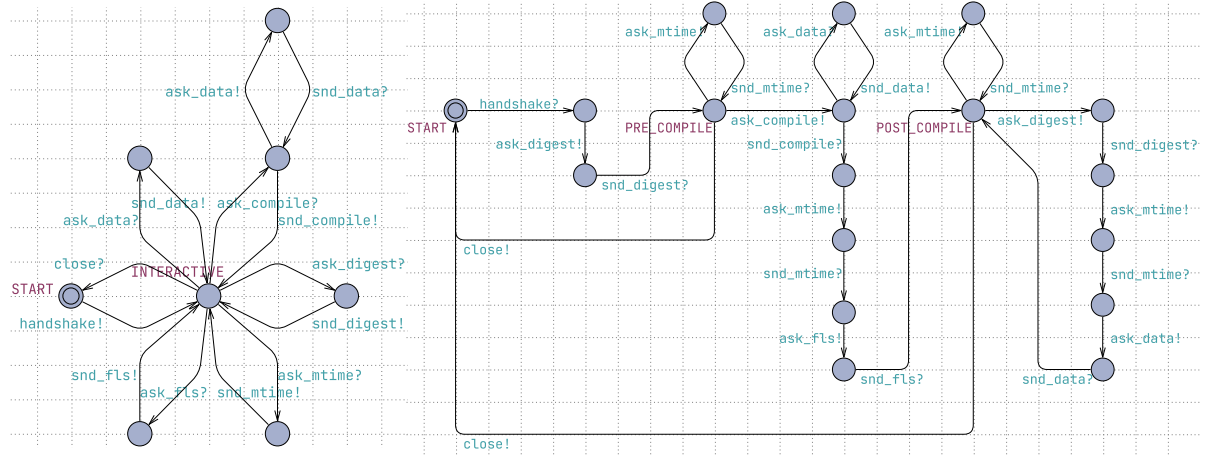
Architecture › LaTeX instrumentation



Architecture › LaTeX instrumentation



Architecture > Client–server communication



FLSdb

Demo

Future directions

- Internal tooling
- External tooling

Internal tooling

- A dashboard to summarize the benefits of the approach
- Fully traceable and explainable build reports
- Better dependency (and required recompilation) tracking

External tooling

- Automatic deployment and archival of artifacts
- Fine-grained access control

⇝ Towards a fully integrated document management system