

# A deep dive into Lean's processing pipeline

**Sebastian Ullrich**

Lean FRO

INRIA | April 24, 2026

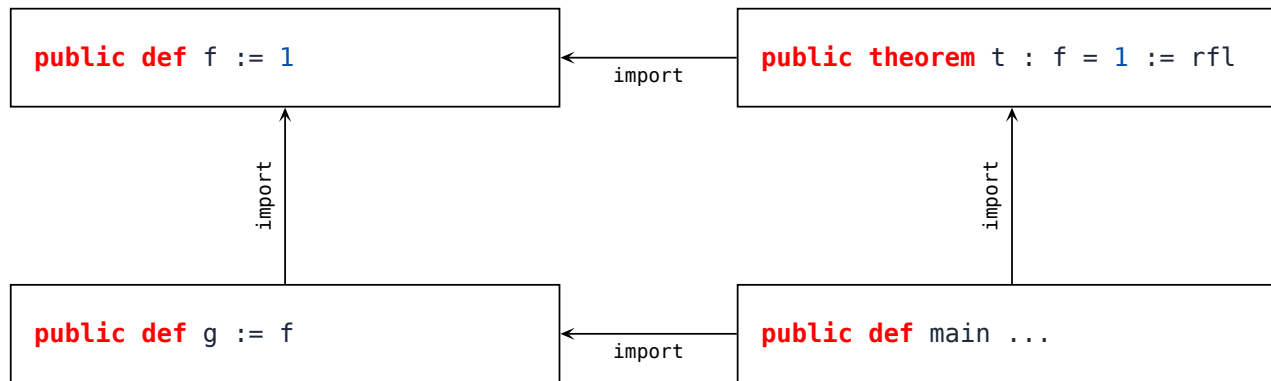


## Layers of processing

- **file level:** Lake, module system
- **declaration level:** incremental parsing & elaboration, parallelism
- **tactic level:** incremental execution



## File level



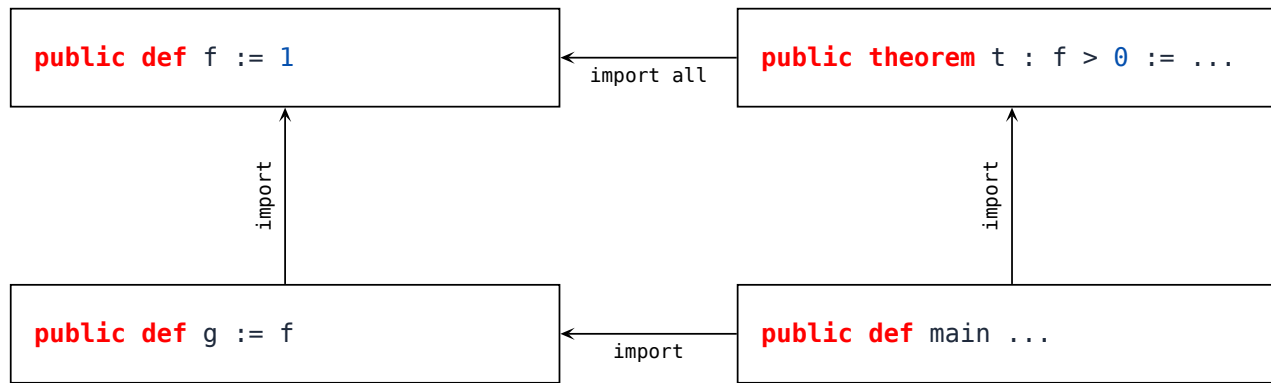
Make-like separate, parallel processing

Can build Mathlib's 9k+ files in 9min30s at 2300% CPU on an AMD EPYC 9455





## File level



**Module system:** consistently separating private and public data to allow for *rebuild avoidance*



## File level

```
GeomSum.lean x
Mathlib > Algebra > Field > GeomSum.lean > {} <section>
60 lemma geom_sum_inv (hx1 : x ≠ 1) (hx0 : x ≠ 0) (n : ℕ) :
61   ∑ i ∈ range n, x⁻¹ ^ i = (x - 1)⁻¹ * (x - x⁻¹ ^ n * x) :
62   have h₁ : x⁻¹ ≠ 1 := by rwa [inv_eq_one_div, Ne, div_eq_iff]
63   have h₂ : x⁻¹ - 1 ≠ 0 := mt sub_eq_zero.1 h₁
64   have h₃ : x - 1 ≠ 0 := mt sub_eq_zero.1 hx1
65   have h₄ : x * (x ^ n)⁻¹ = (x ^ n)⁻¹ * x :=
66     Nat.recOn n (by simp) fun n h => by
67       rw [pow_succ', mul_inv_rev, ← mul_assoc, h, mul_assoc,
68         inv_mul_cancel₀ hx0]
69   rw [geom_sum_eq h₁, div_eq_iff_mul_eq h₂, ← mul_right_inj',
70     mul_inv_cancel₀ h₃]
71   simp [mul_add, add_mul, mul_inv_cancel₀ hx0, mul_assoc, h₄,
72     add_left_comm]
73   rw [add_comm _ (-x), add_assoc, add_assoc _ _ 1]
74
```

```
bash x
kim@chonk:~/mathlib4-2$
```



## Module system basics

Opt-in via `module` header keyword

- Declarations *and imports* now `private` by default; adjust using `public`
- `def` bodies are also private to the module unless `@[expose]`
- Proofs are always private
- All metaprograms need to be annotated/imported with `meta`

=> Much more precise control over what information a module exposes

More info: Reference Manual §5.4+, Lean Hackathon keynote #2, Lean Together 2026 talk



## Basic module system optimizations

- Rebuild avoidance: changes to private data stop at the end of the file; public changes at the next `import`
- 64% of mathlib4 data is private; RAM use of `import Mathlib` reduced by 48%
- TBD: download only necessary parts from cloud cache
- Do not link `meta` code into final executable
- potentially: start downstream builds as soon as public data is ready

Lake is being extended with a *content-addressed* local/remote cache to allow for cache hits even when transitive inputs are not identical



## Module system: implementation (difficulties)

Primary goals: no kernel changes, minimize metaprogramming API changes

```
public structure Kernel.Environment where
  ...

structure VisibilityMap (α : Type) where
  private : α
  public  : α

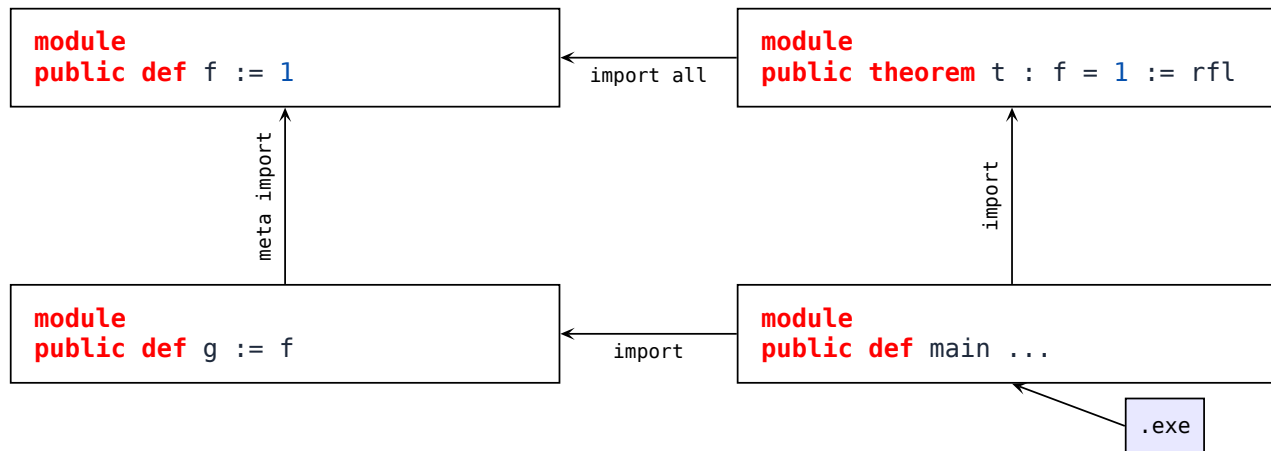
public structure Environment where
  isExporting : Bool
  private base : VisibilityMap Kernel.Environment
  ...

def Environment.find? (env : Environment) (n : Name) : Option ConstantInfo :=
  if env.isExporting then ...
```

*Environment extensions* gained optional callback to separate private from public export data



## In progress: separate compilation



The compiler both creates less-stable public data and is only relevant for downstream `meta` code

=> move to separate build step



## Declaration level: parallelism

- Theorem bodies (proofs)
- Kernel checking
- Linting
- (Meta) compilation

Diagnostics are collected in an  
asynchronously growing tree

Cmdline reports pre-order traversal of  
tree, language server unordered traversal

```
tests > scratch.lean > ...
1  theorem t1 : False := by unsolved goals<math>\Leftarrow\vdash</math> False
2  sleep 1100
3  sleep 1100
4
5  theorem t2 : False := by unsolved goals<math>\Leftarrow\vdash</math> False
6  sleep 1000
7  sleep 1000
8
9  theorem t3 : False := by unsolved goals<math>\Leftarrow\vdash</math> False
10 sleep 1300
11 sleep 1300
12
13 theorem t4 : False := by unsolved goals<math>\Leftarrow\vdash</math> False
14 sleep 1200
15 sleep 1200
16
```



## Parallelism: implementation (difficulties)

```
structure AsyncConstantInfo where
  name      : Name
  kind      : ConstantKind
  sig       : Task ConstantVal
  constInfo : Task ConstantInfo

public structure Environment where
  ...
  checked           : Task Kernel.Environment
  -- (simplified)
  private asyncConstsMap : VisibilityMap (NameMap AsyncConstantInfo)
  ...

def Environment.findTask (env : Environment) (n : Name) : Task (Option AsyncConstantInfo)
def Environment.findConstVal? (env : Environment) (n : Name) : Option ConstantVal
def Environment.find? (env : Environment) (n : Name) : Option ConstantInfo
```

Environment extensions again add extra (back-compat) complexity, now come with an **asyncMode** defaulting to "main thread only".



# trace.profiler





## Declaration & tactic level: incrementality

Purely syntactic, for now: reuse parser & elaborator results strictly above change

Tactic combinators can opt into nested incrementality support

In progress: also allow saving this data to disk

```
346 theorem Stmt.simplify_correct (h : (σ, s) ↓ σ') : (σ, s.simplify) ↓ σ' := by
347   induction h with
348   | skip => exact Bigstep.skip
349   | seq h1 h2 ih1 ih2 => exact Bigstep.seq ih1 ih2
350   | assign => apply Bigstep.assign; simp [*]
351   | whileTrue heq h1 h2 ih1 ih2 =>
352     sleep 200
353     simp_all
354     rw [← Expr.eval_simplify] at heq
355     split
356     next h => rw [h] at heq; simp at heq
357     next hnp => simp [hnp] at ih2; apply Bigstep.whileTrue heq ih1 ih2
358   | whileFalse heq =>
359     sleep 200
360     simp_all
361     split
362     next => exact Bigstep.skip
363     next => apply Bigstep.whileFalse; simp [heq]
364   | iffFalse heq h ih =>
365     sleep 200
366     simp_all
367     rw [← Expr.eval_simplify] at heq
368     split <;> simp_all
369     rw [← Expr.eval_simplify] at heq
370     apply Bigstep.iffFalse heq ih
371   | iffTrue heq h ih =>
372     sleep 200
373     simp_all
374     rw [← Expr.eval_simplify] at heq
375     split <;> simp_all
376     rw [← Expr.eval_simplify] at heq
377     apply Bigstep.iffTrue heq ih
378
```



**Thank You**

Questions?

Speaker notes