

Unicorn

/ˈjuːnɪkɔːn/

“The unicorn is a legendary animal from European folklore that resembles a white horse with a large, pointed, spiraling horn projecting from its forehead”

- Wikipedia



Unicorn

/ˈjuːnɪkɔːn/

“The unicorn is a legendary animal from European folklore that resembles a white horse with a large, pointed, spiraling horn projecting from its forehead.”

wrong!

- Wikipedia



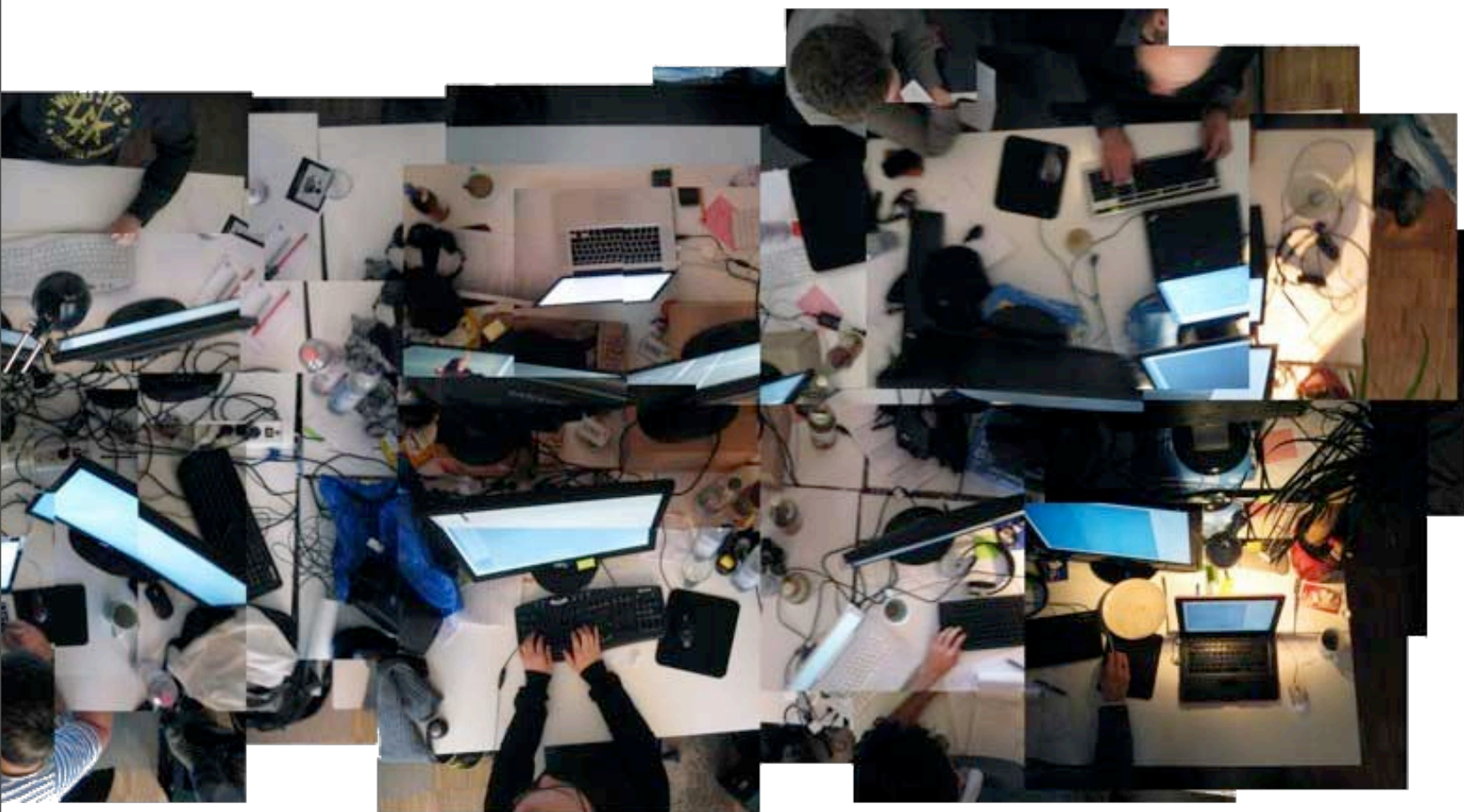
Unicorn

/ˈjuːnɪkɔːn/

“Unicorn is a single-threaded HTTP server for Ruby applications designed to only serve fast clients on low-latency, high-bandwidth connections”

- Bogomips





Grzegorz Witek

Software Developer at SponsorPay
@arnvald

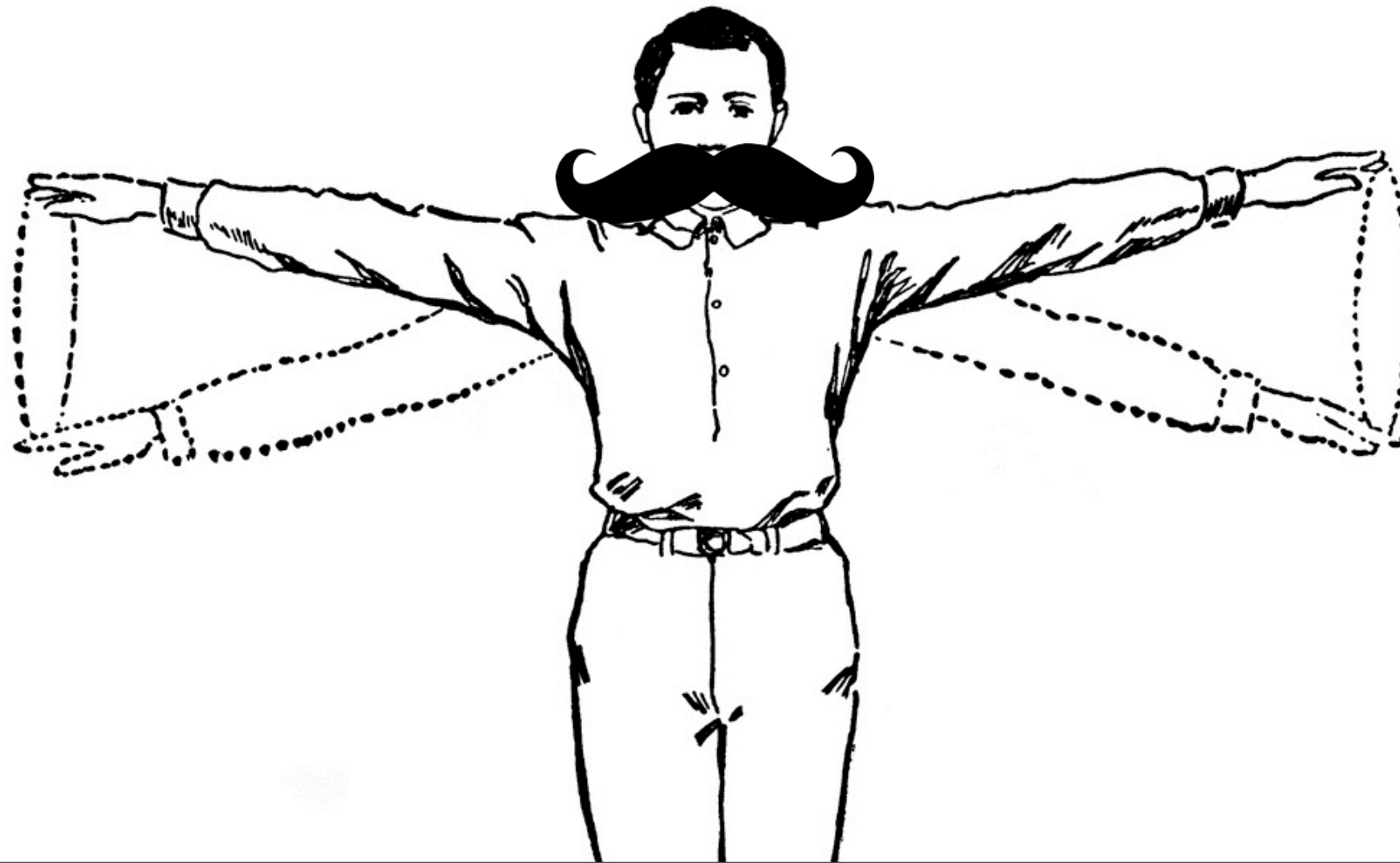
Simon Kröger

VP Engineering at SponsorPay
@kroegerberlin

...but we had to kill unicorns

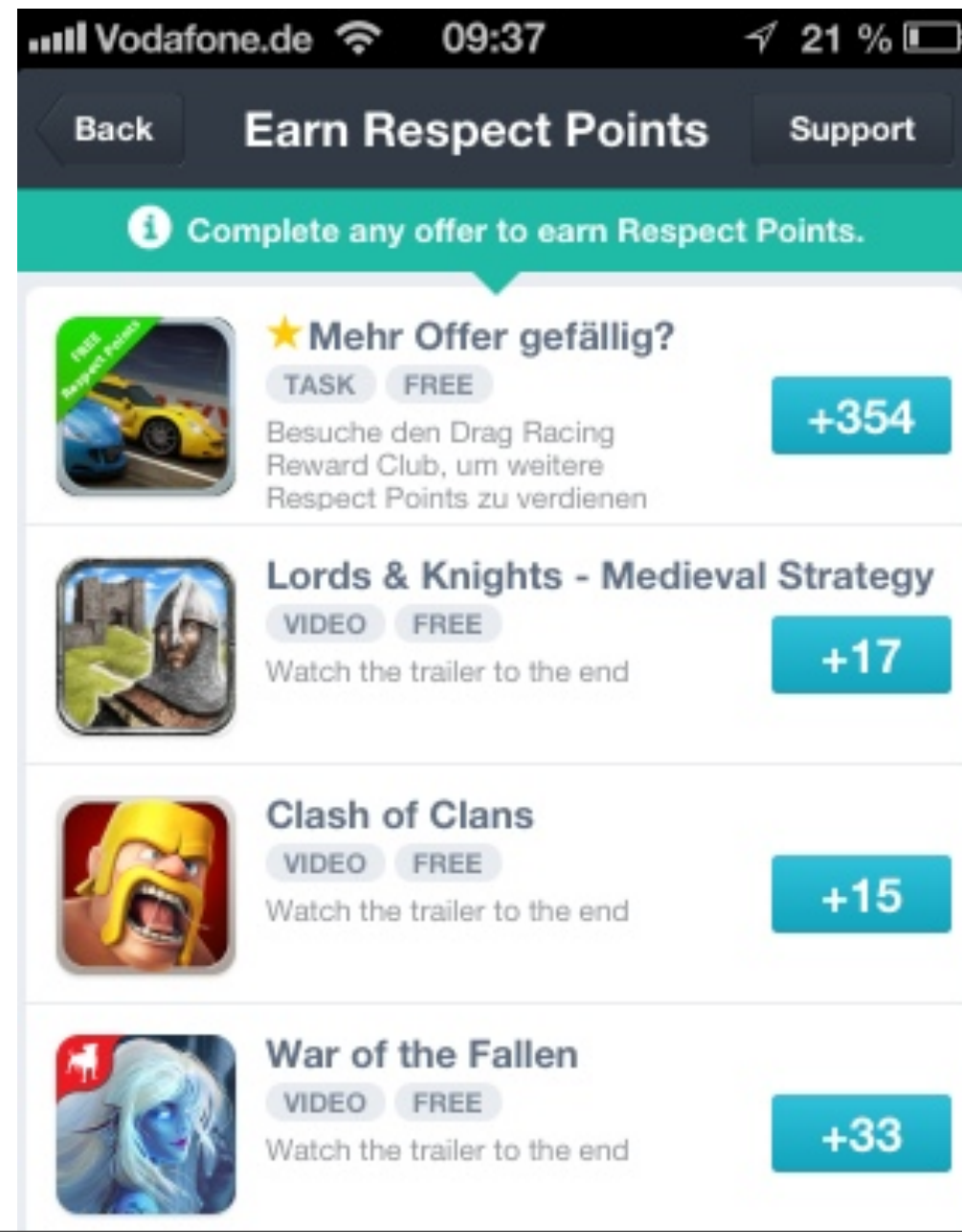
we are not hiring!
unless you are great 'cos in that case we should talk :)

SponsorPay™

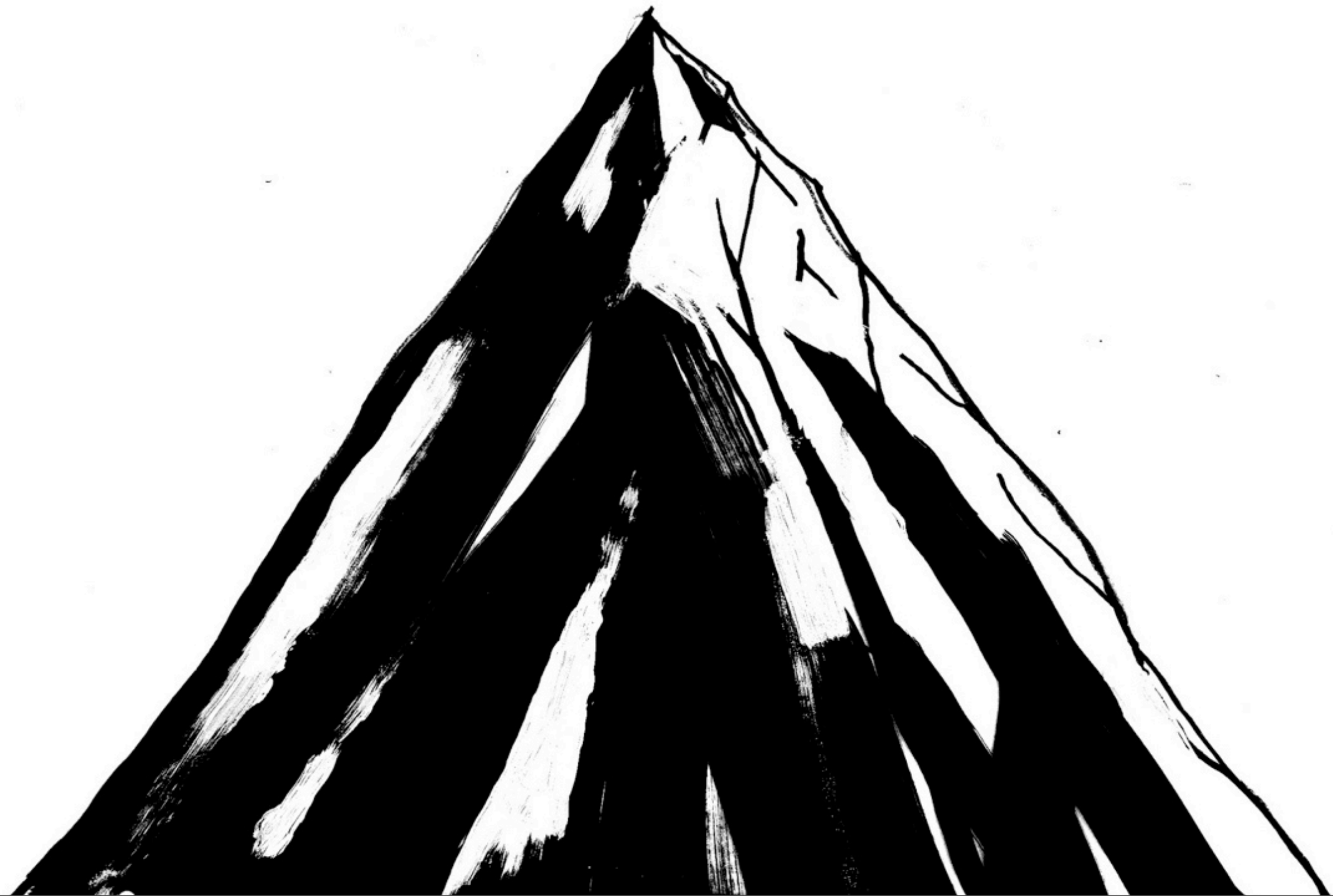


The App

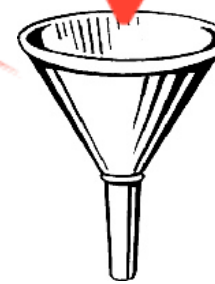
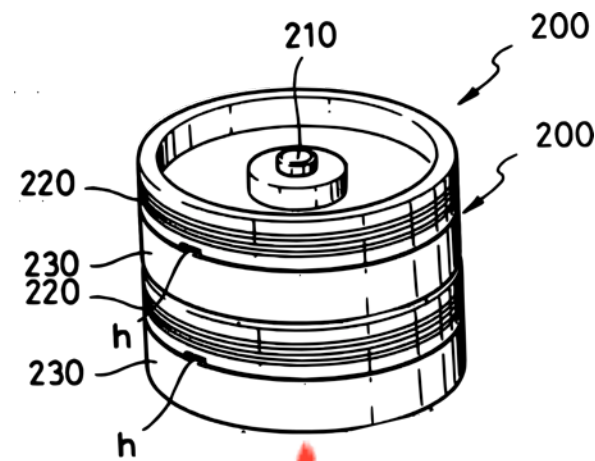
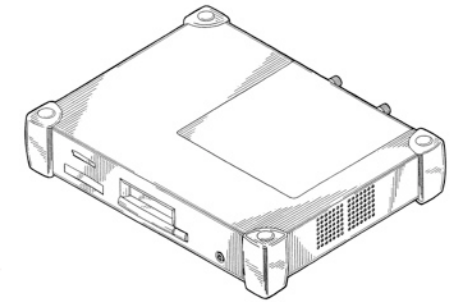
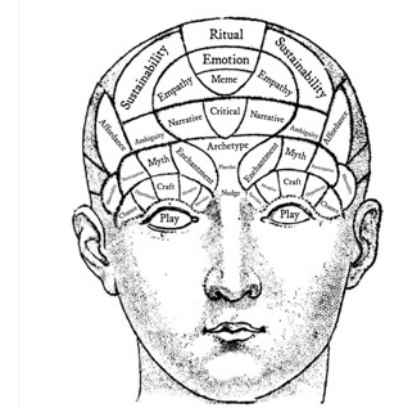
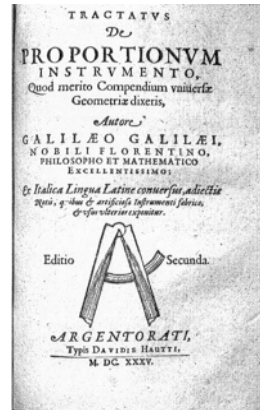
integrated inside a game



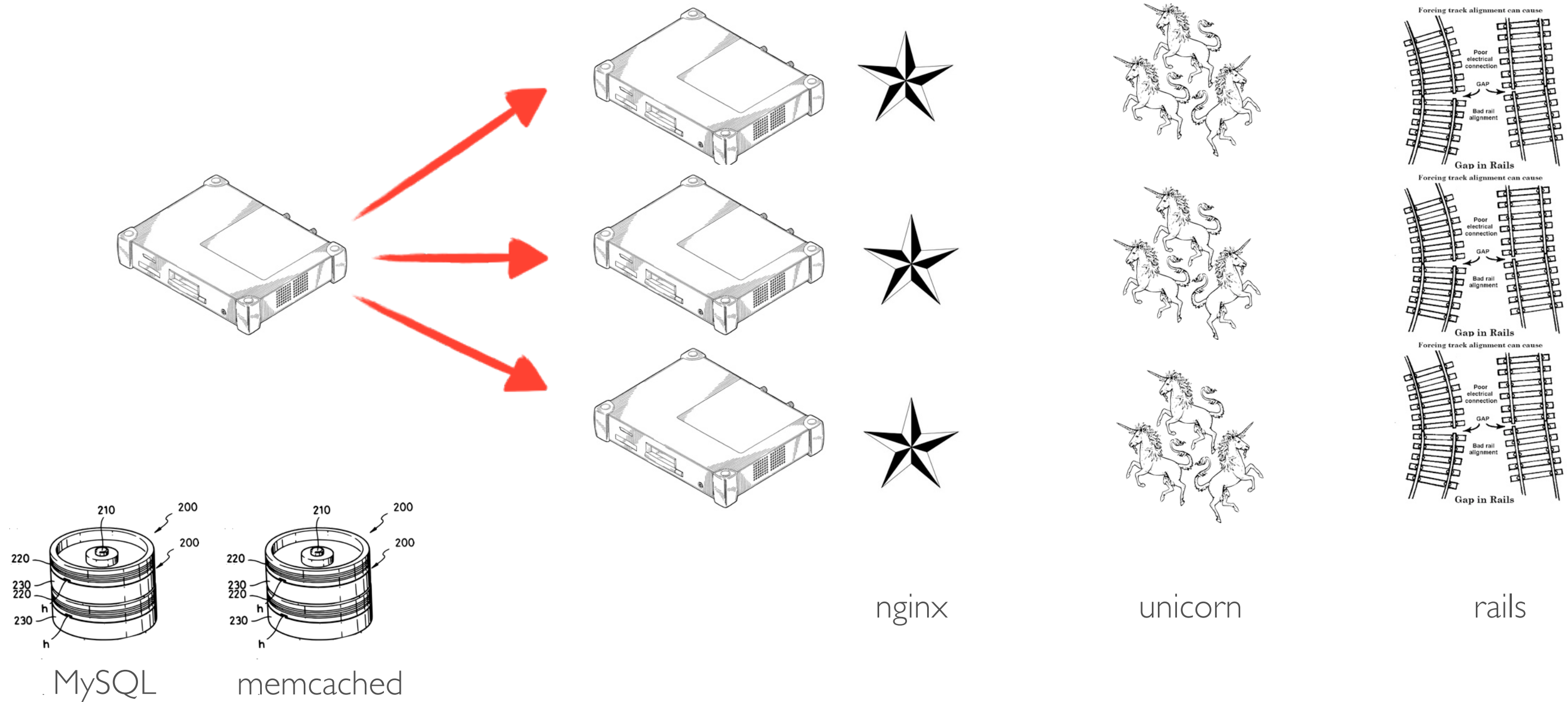
6000m above sea level



Old Flow



Typical architecture

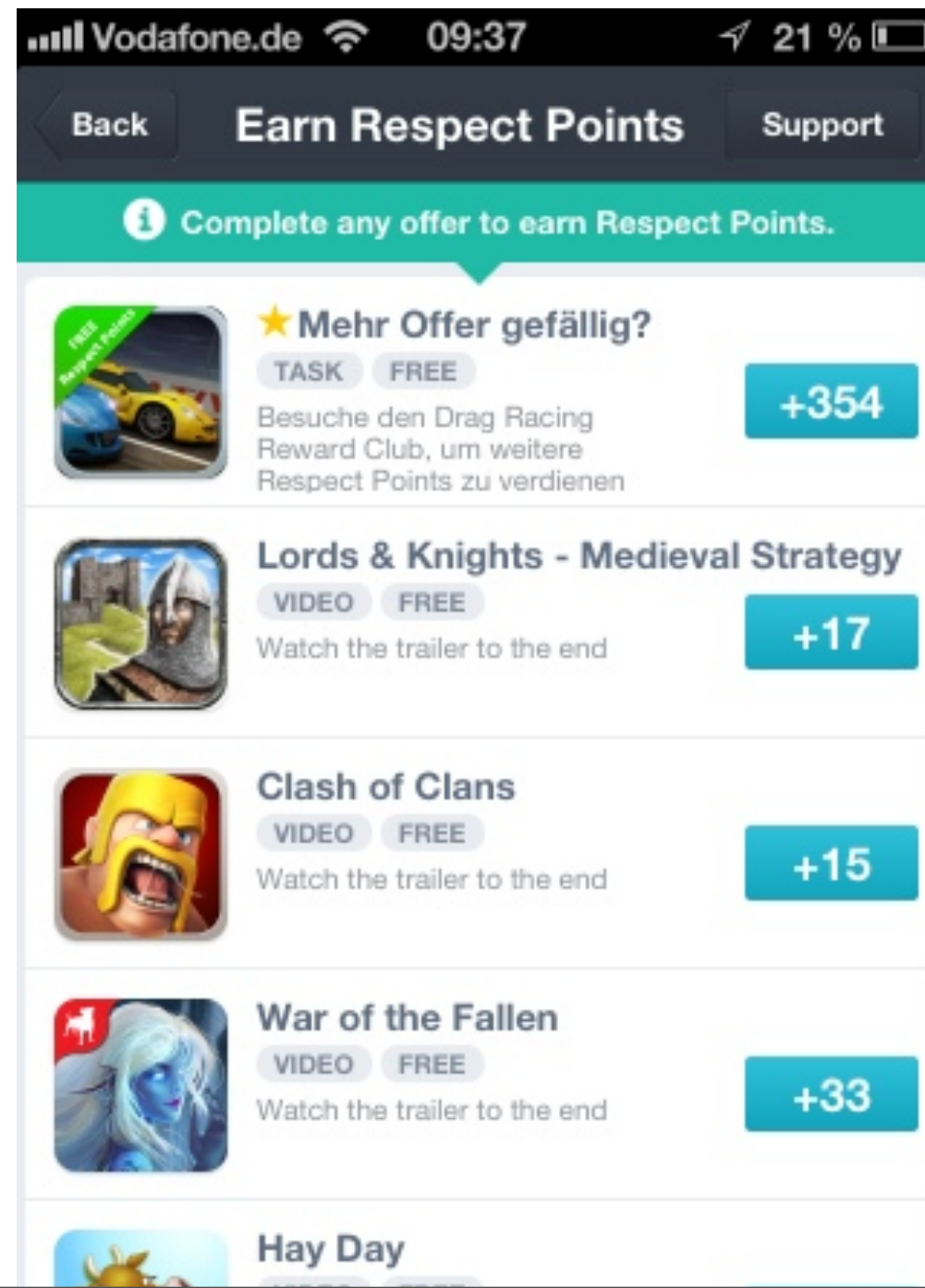


Product Managers came and started playing with our platform

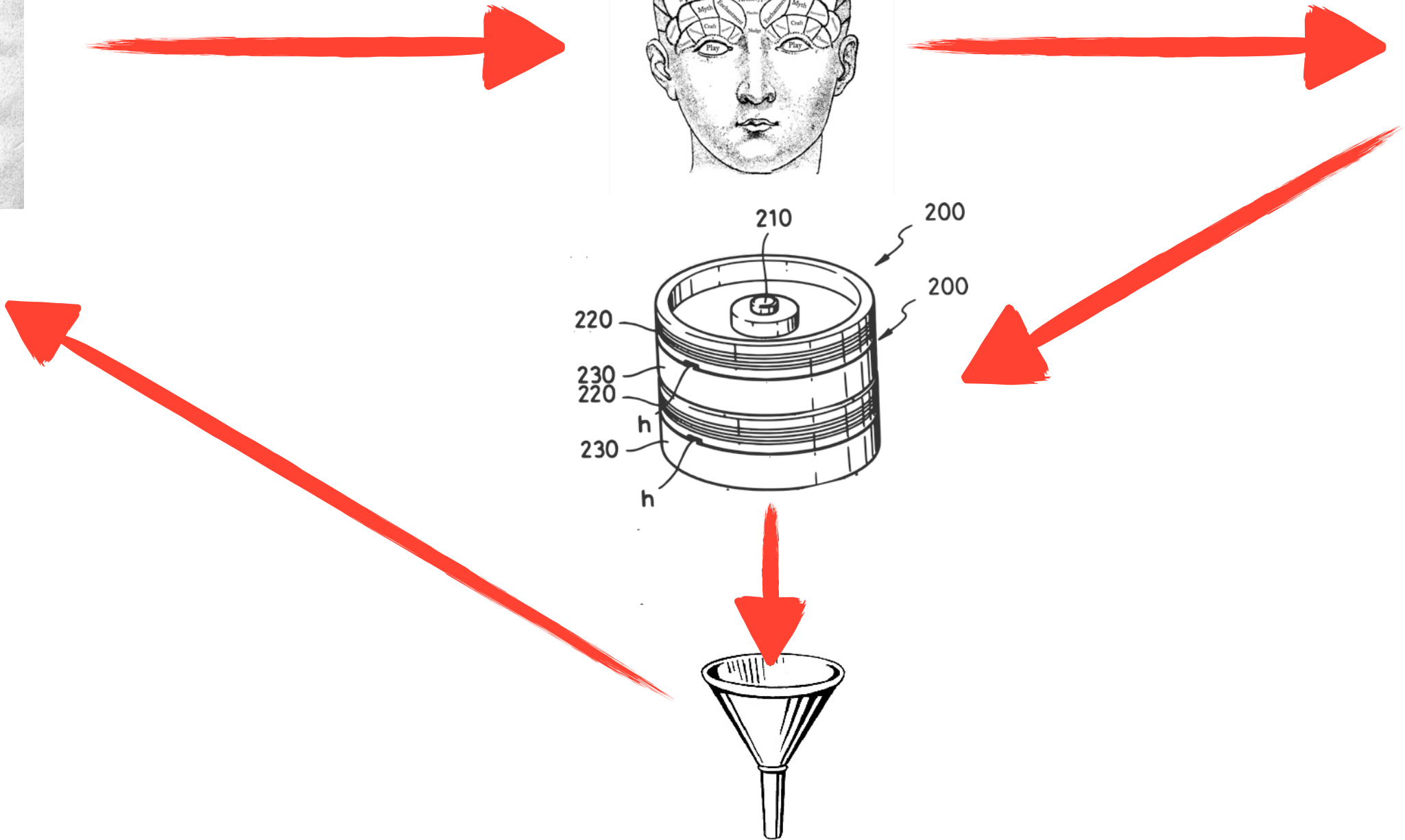
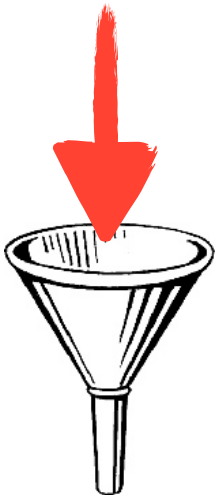
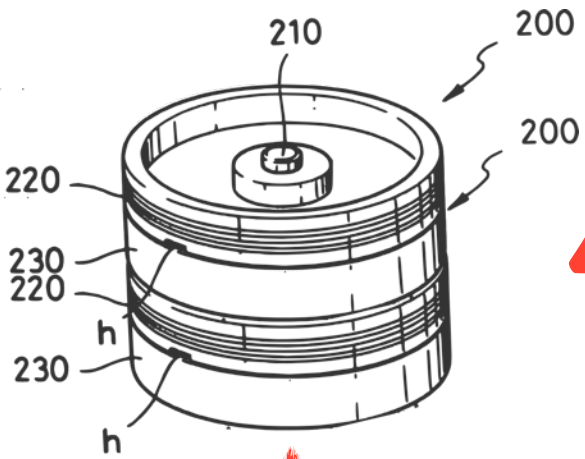
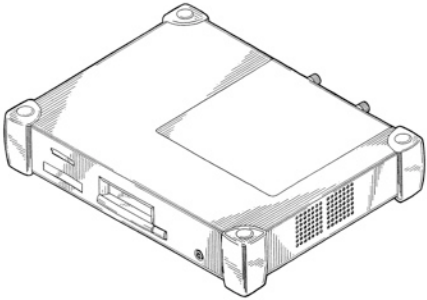
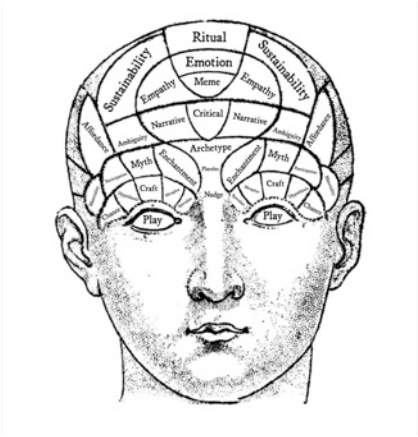
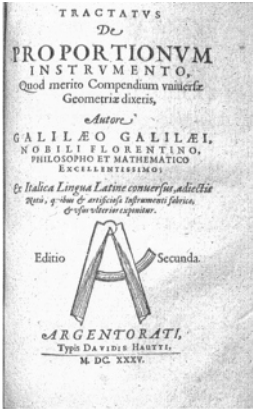


The feature

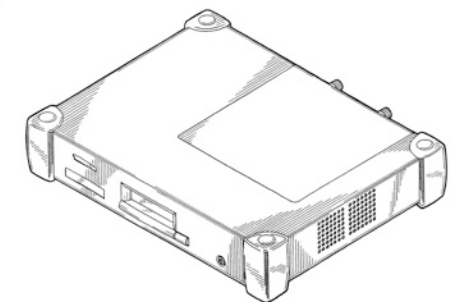
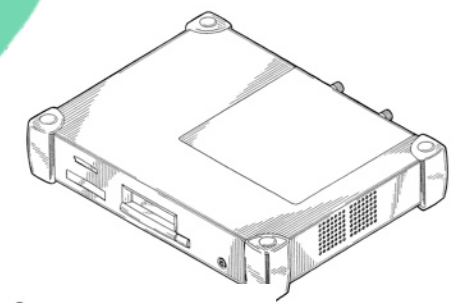
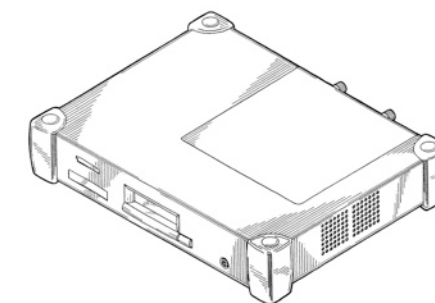
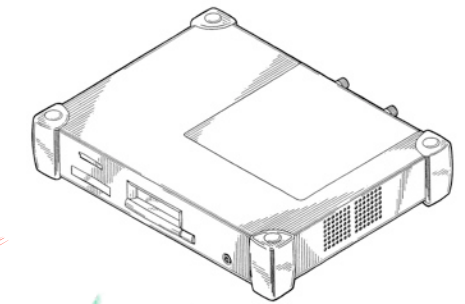
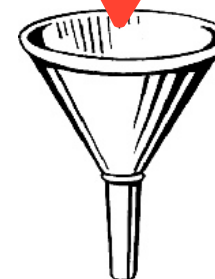
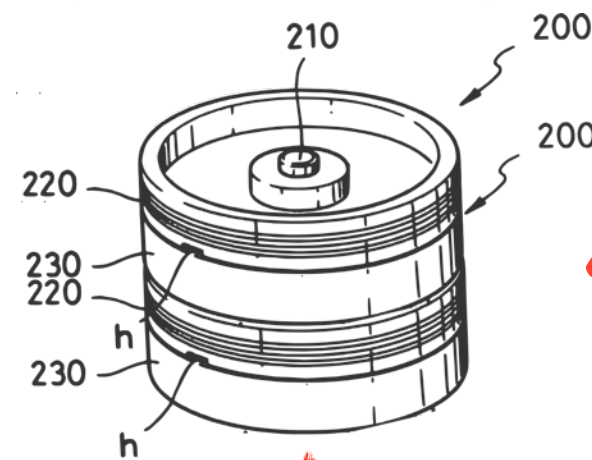
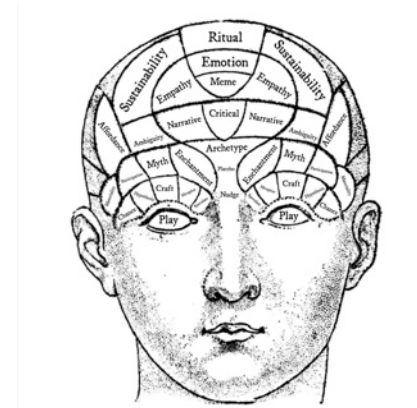
Aggregate offers in real time from different sources



Old Flow



New Flow

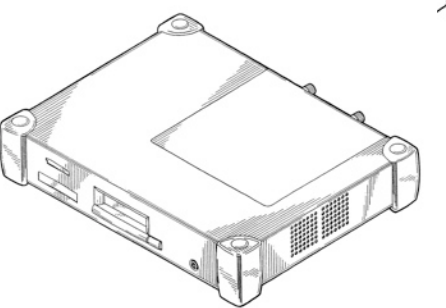
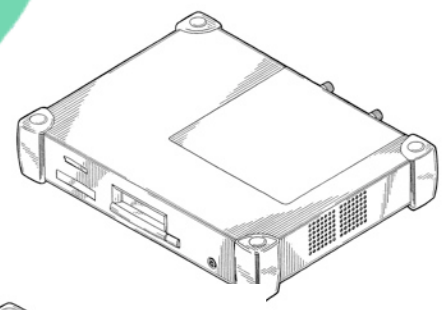
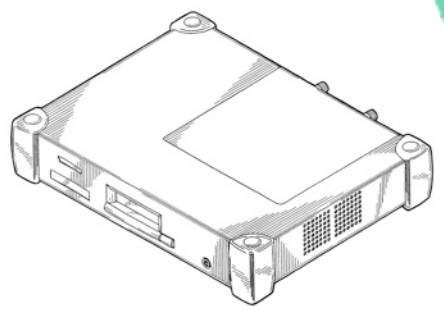
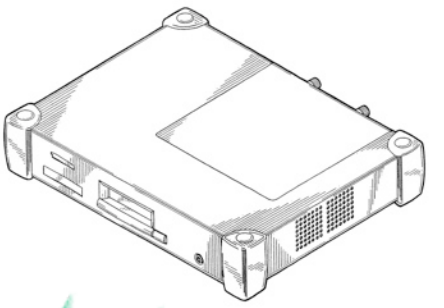
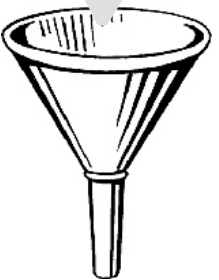
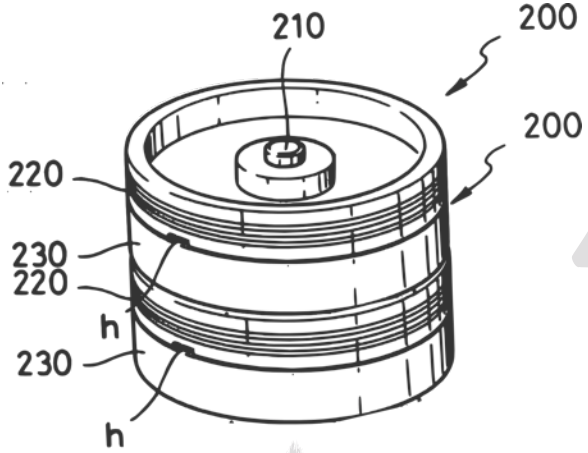
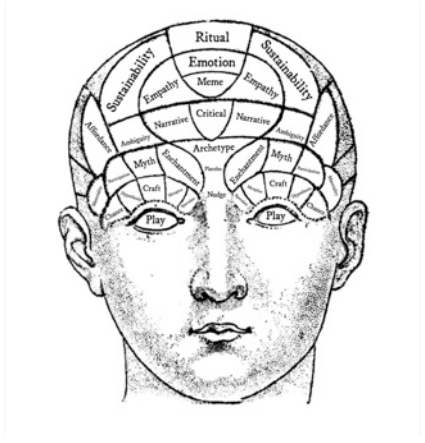
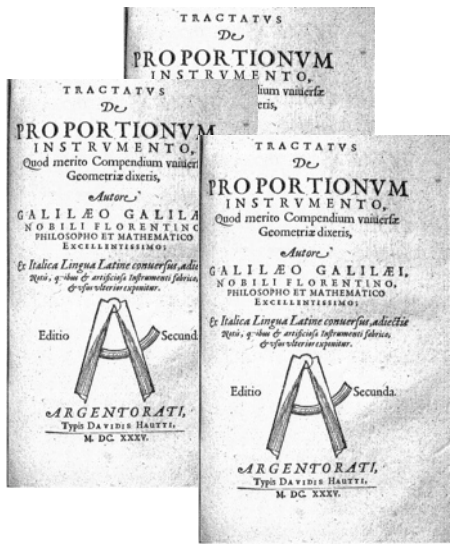


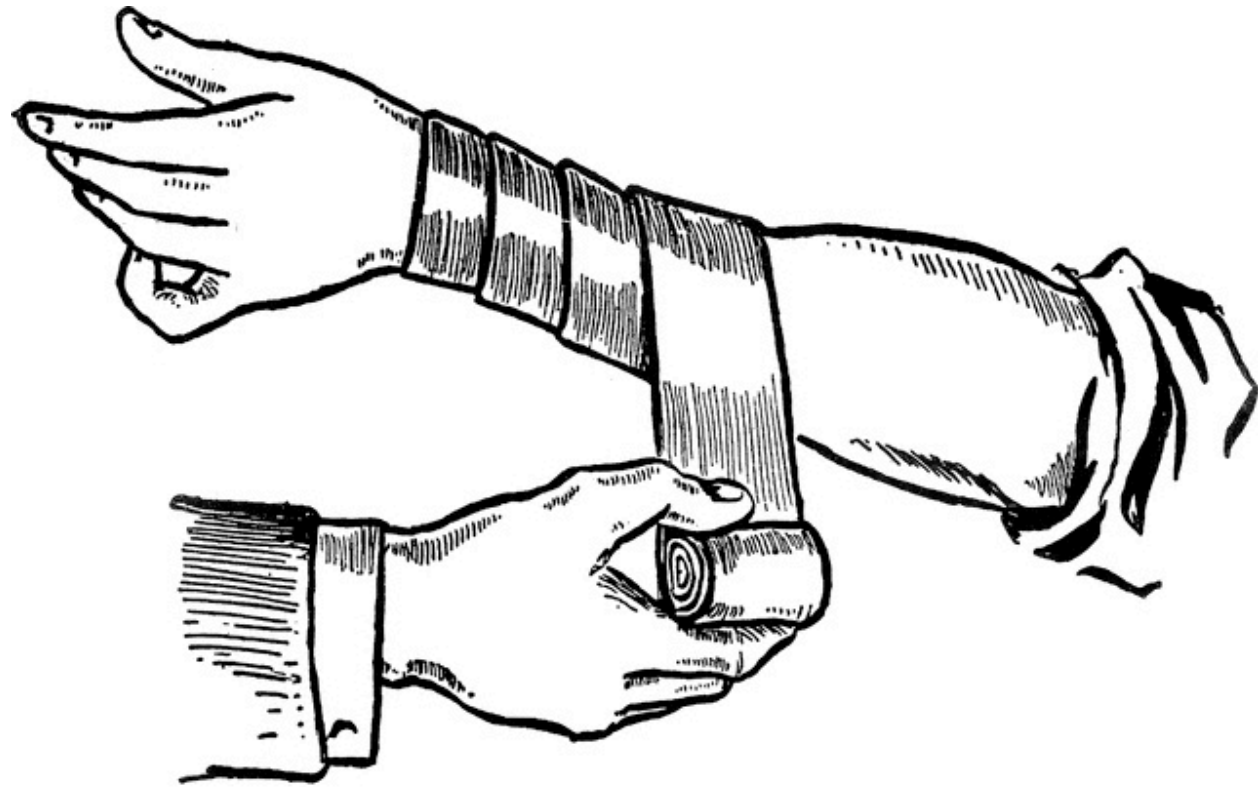
Problem 1

concurrency



Old Flow





libcurl wrappers

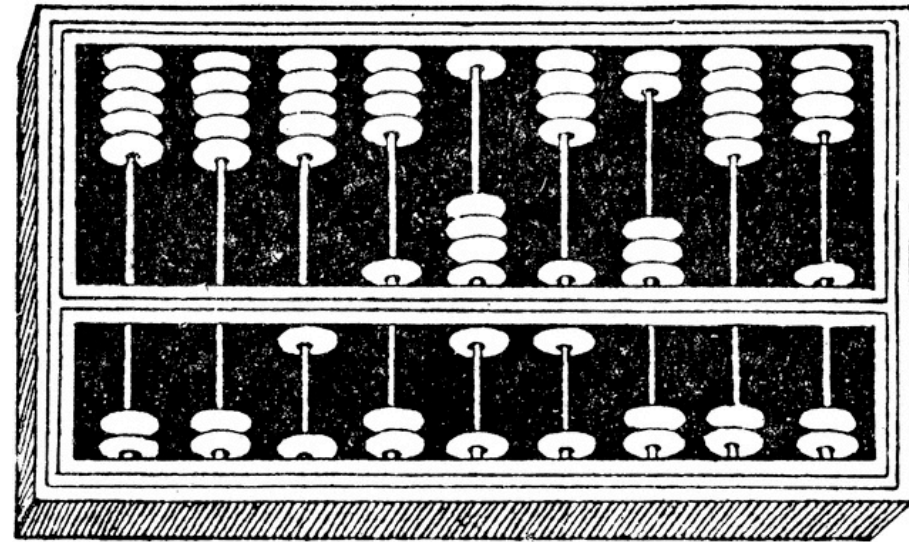
Typhoeus

Curb

Problem 2


maths

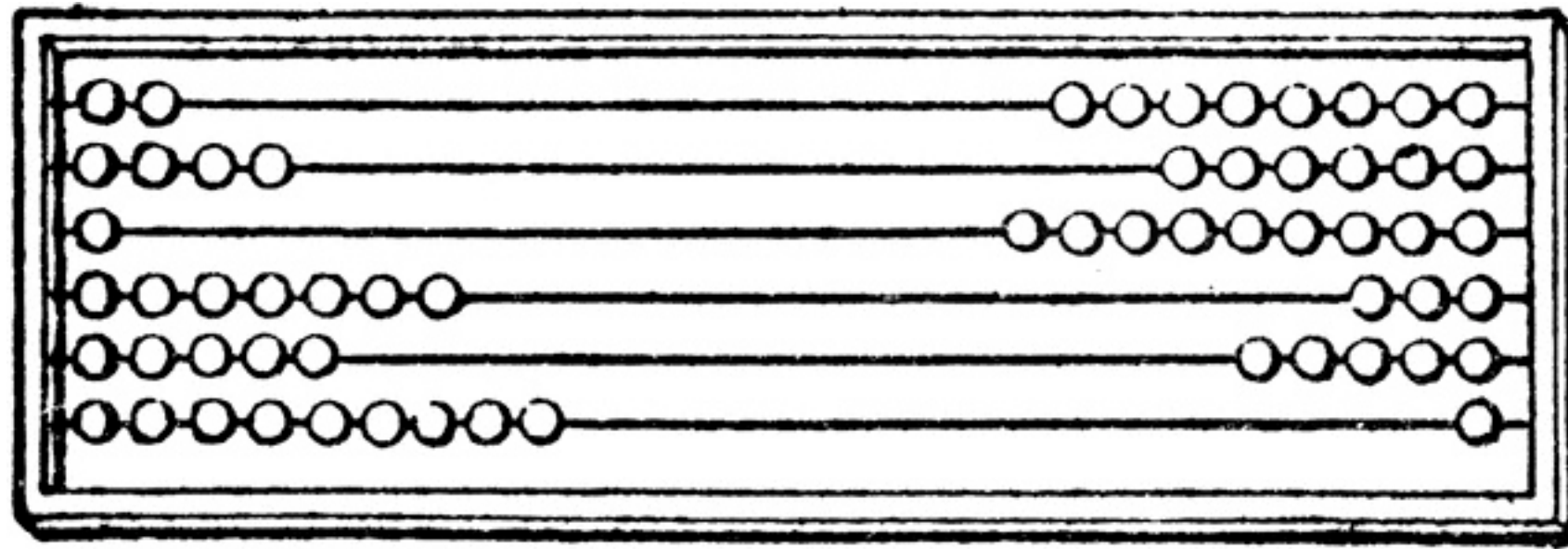




$3000 \text{ connections/s} \times 1 \text{ s} = 3000 \text{ parallel connections}$

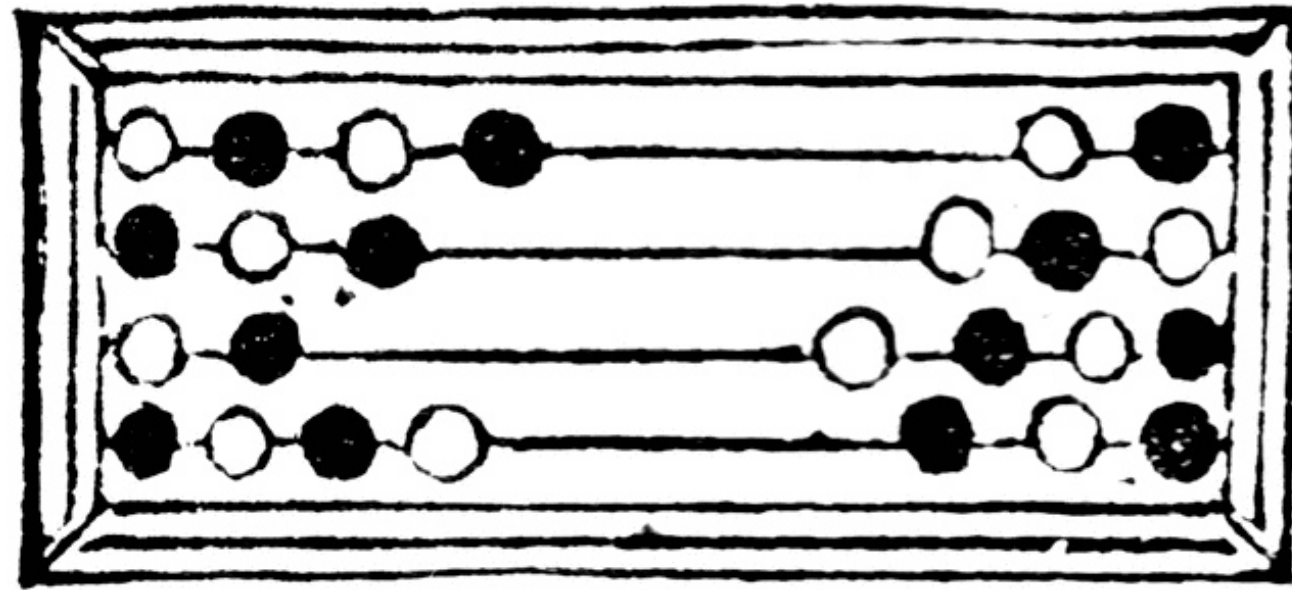
$3000 \text{ connections} / 10 \text{ servers} = 300 \text{ parallel connections/server}$





300 connections = 300 processes

$$300 \times 0.33\text{GB} = 100\text{GB}$$

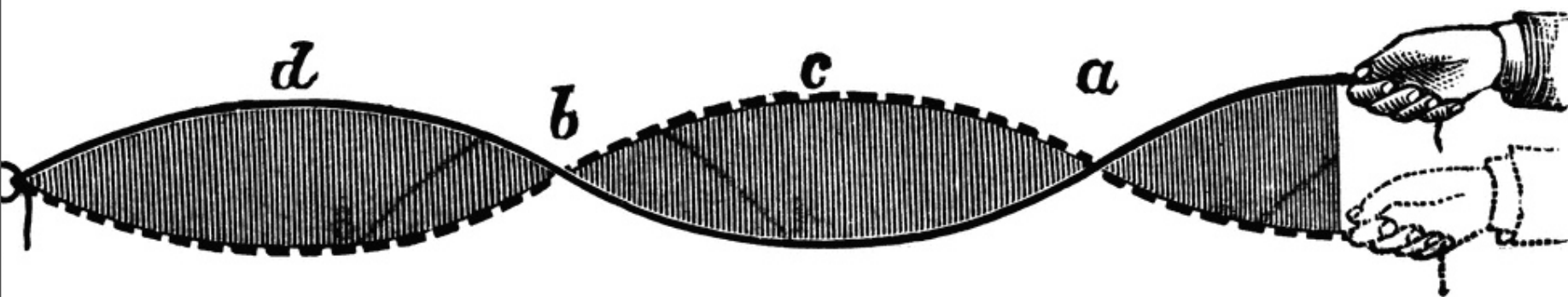


300 processes = 100GB RAM per server

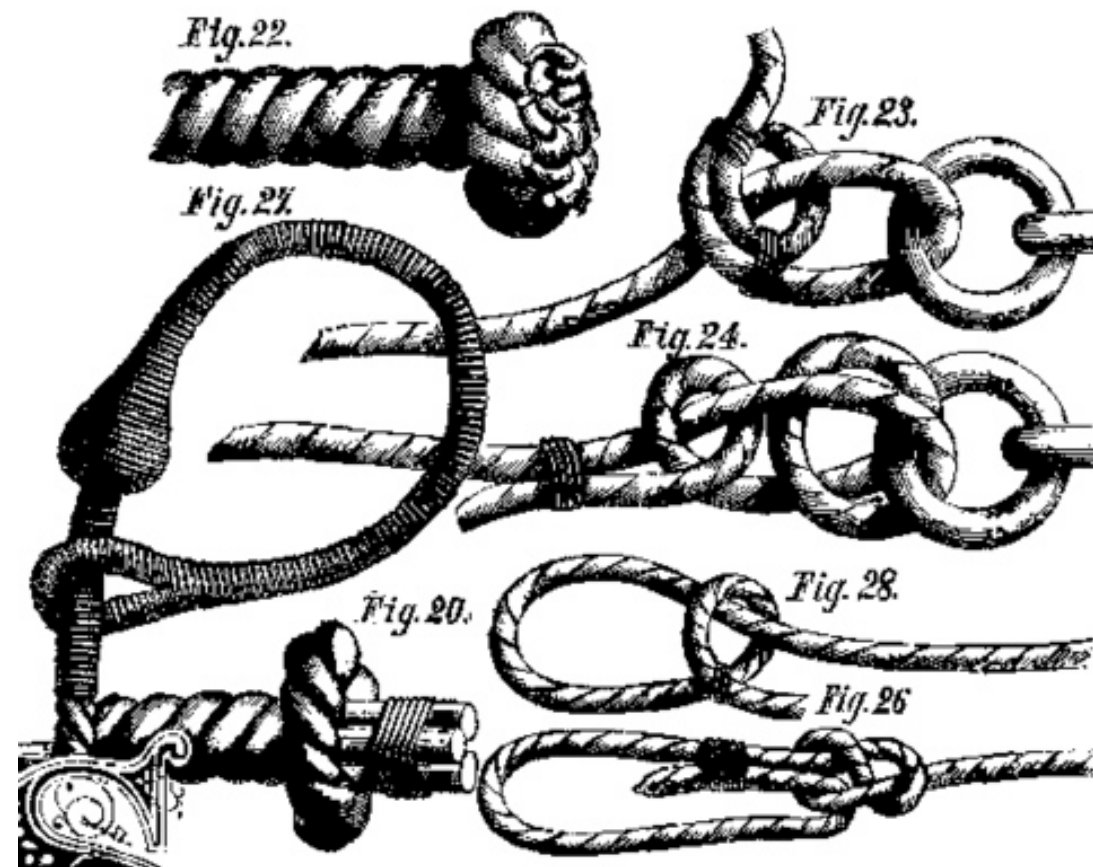
3000 processes = 3000 DB connections

and we are growing!





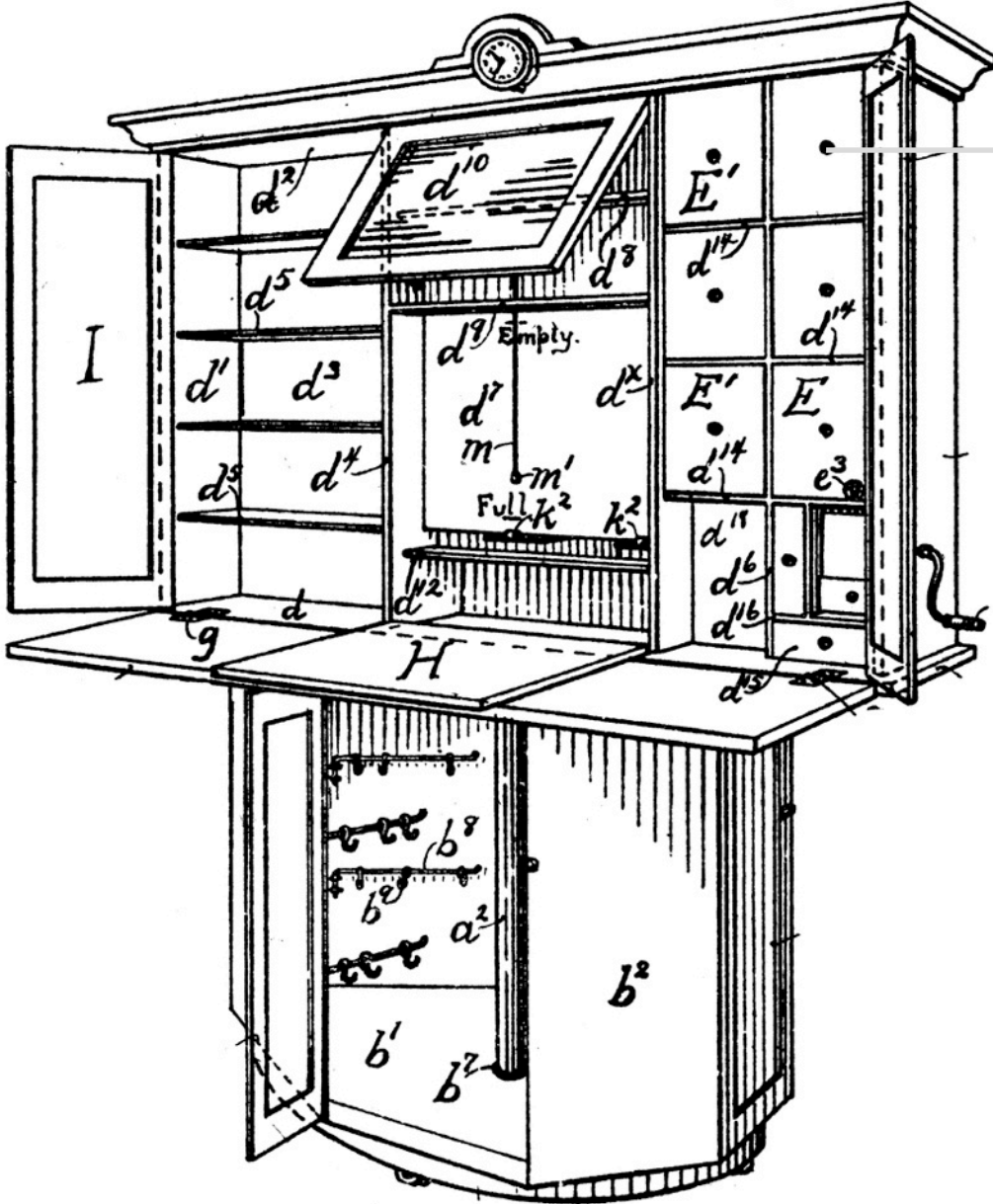
MRI + Threads



Global VM Lock
Global Interpreter Lock

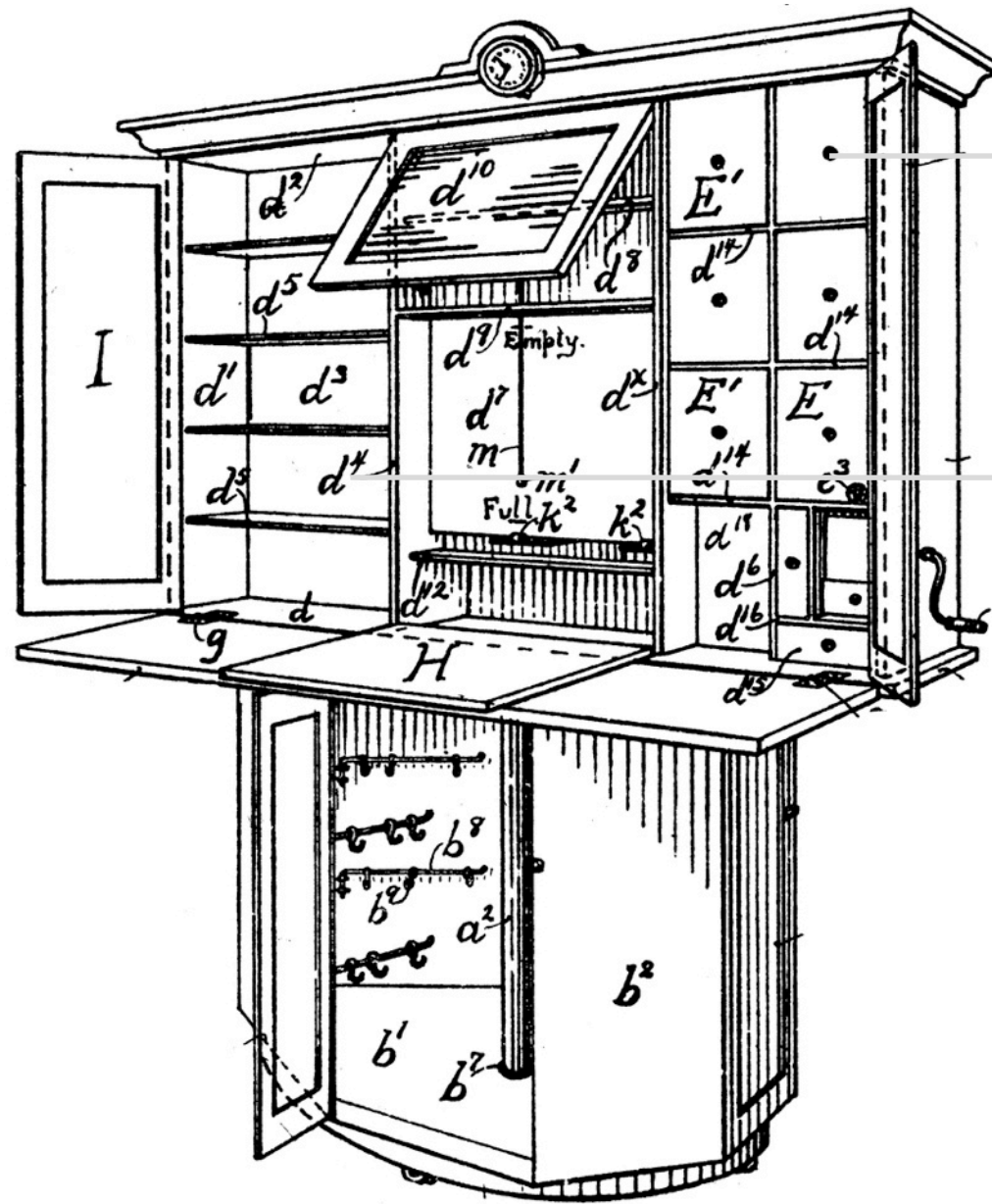
No parallelism

solutions



jRuby

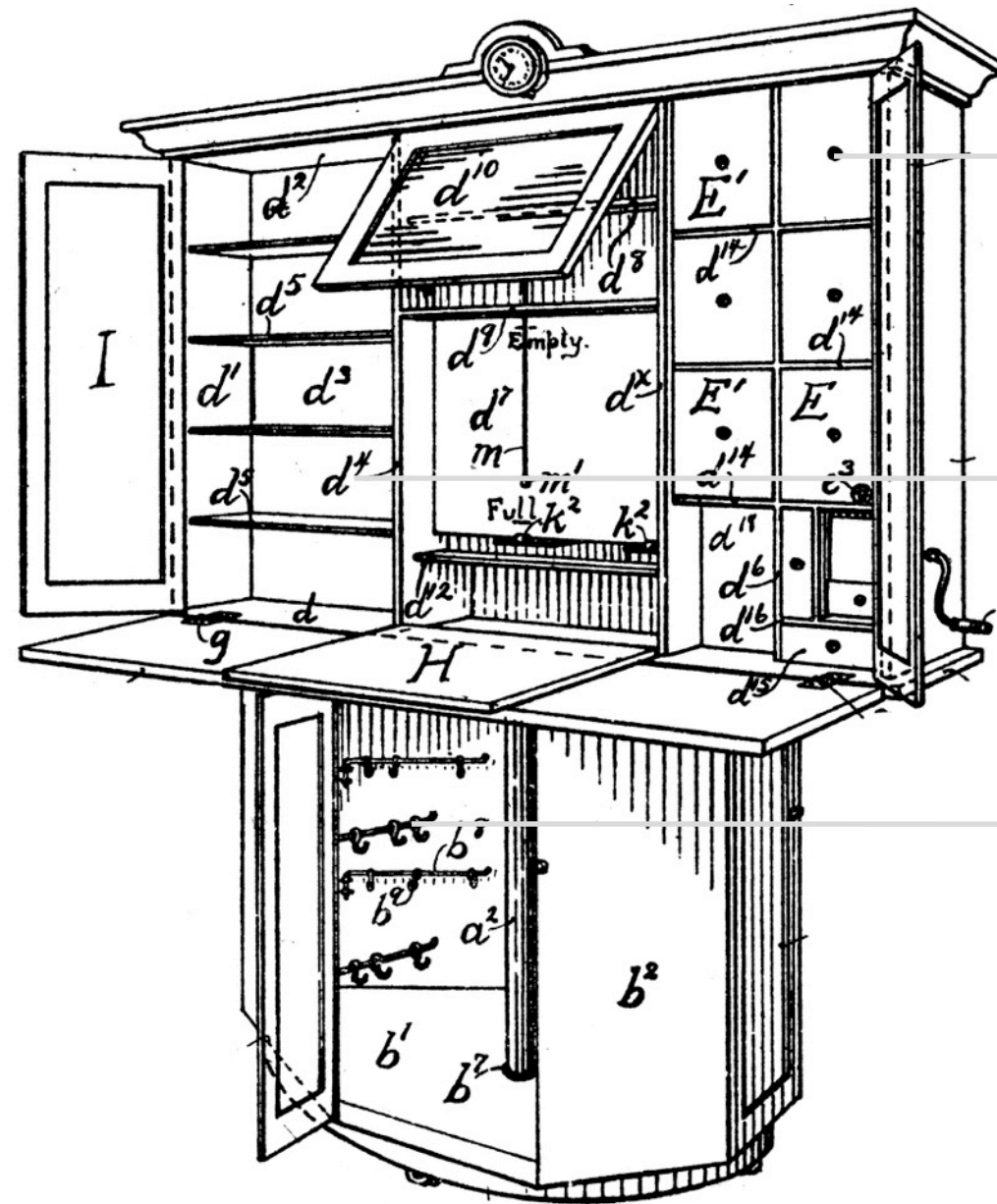
solutions



jRuby

Events

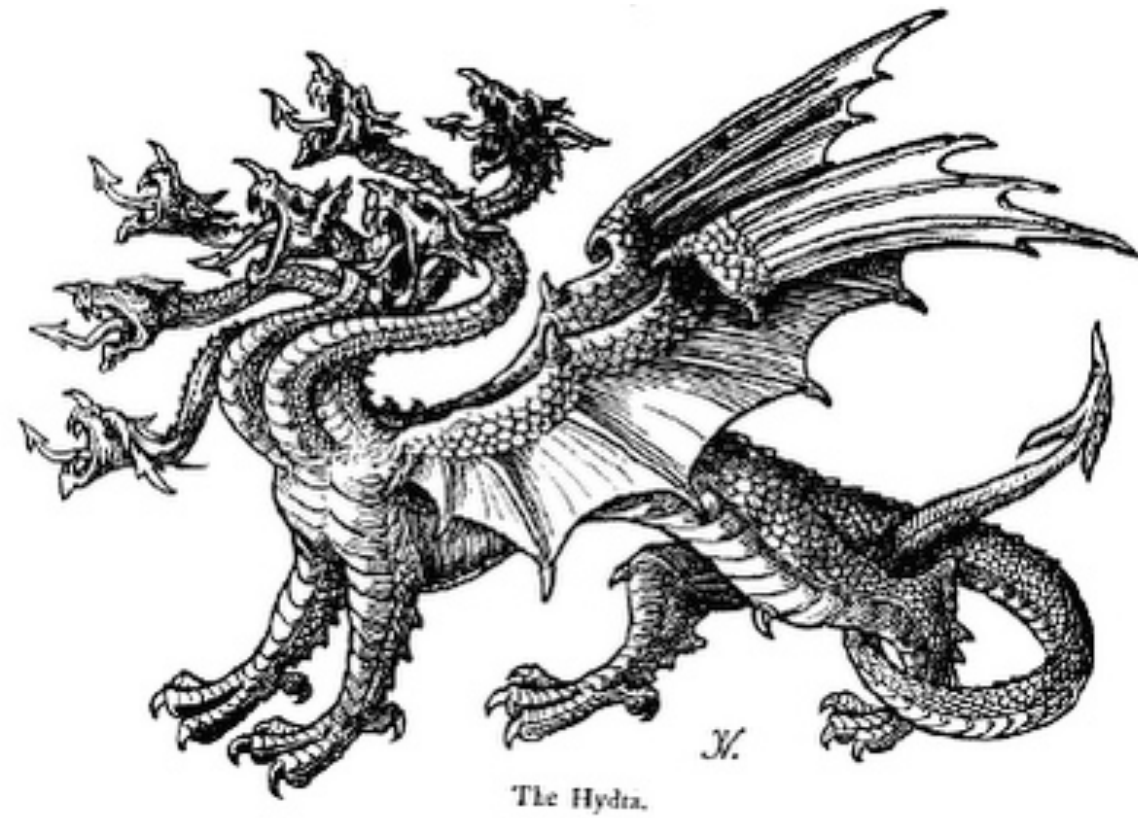
solutions



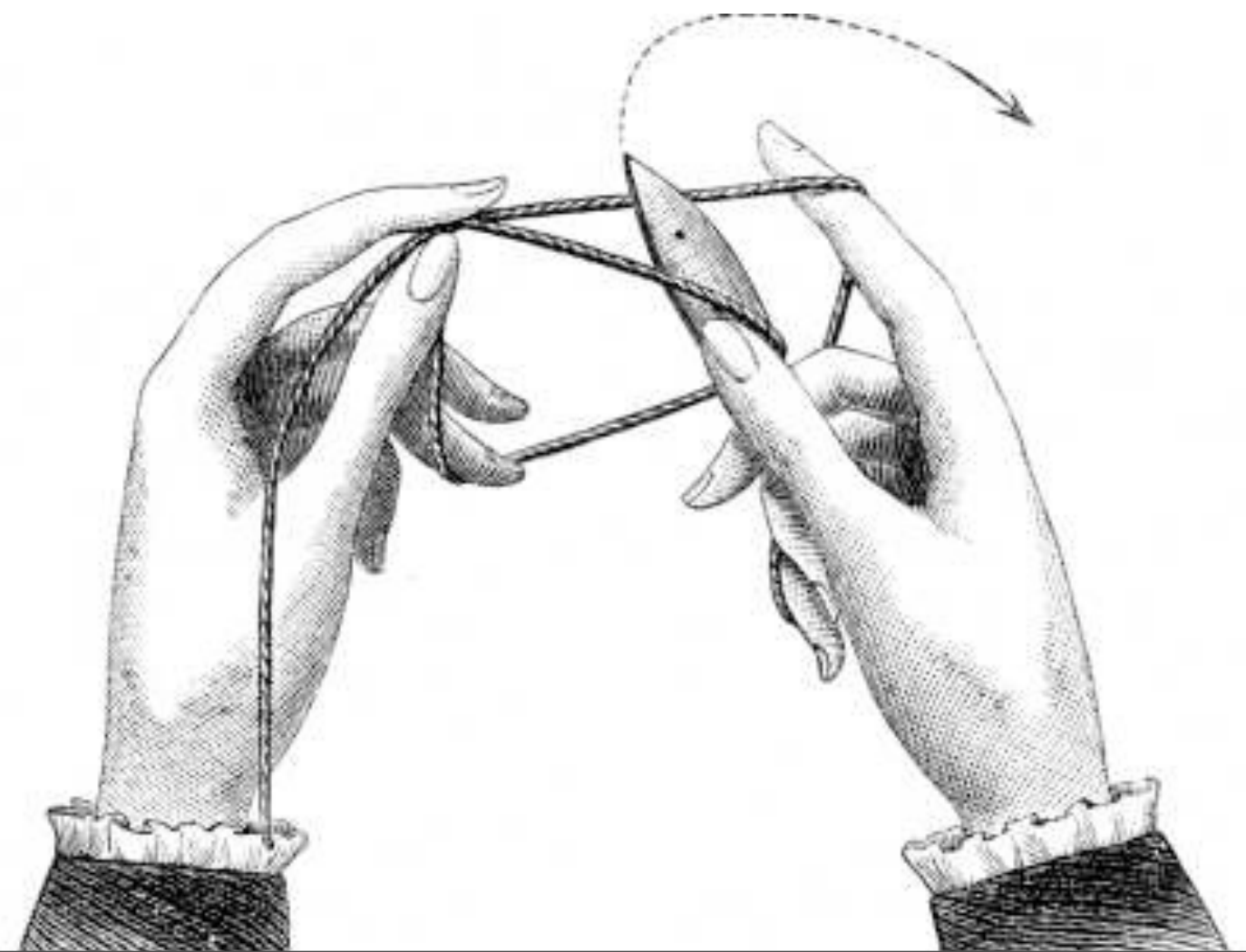
Events

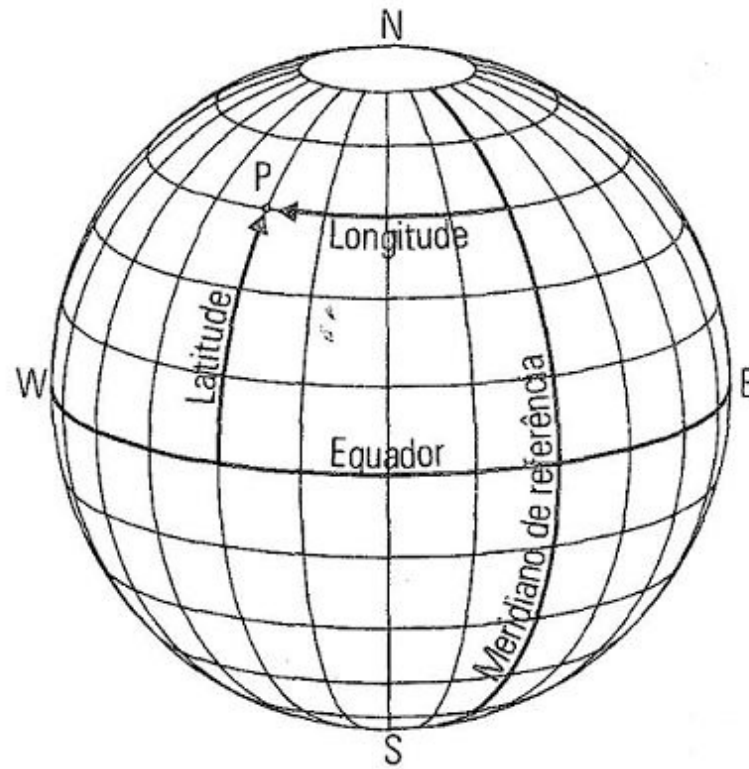
New Brave World

plain threads



MRI + Threads
WAT?





release VM lock

manually: C API (like mysql2 gem)

automatically: non-blocking IO

Some people, when confronted with a problem,
think, "*I know, I'll use threads*"

- and then two they hav erpoblesm.



Problem 3

Server

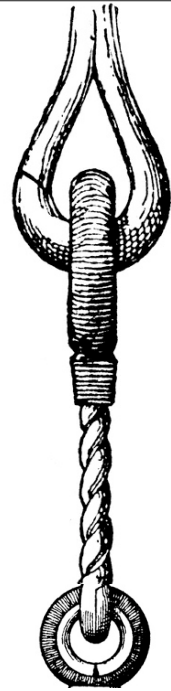




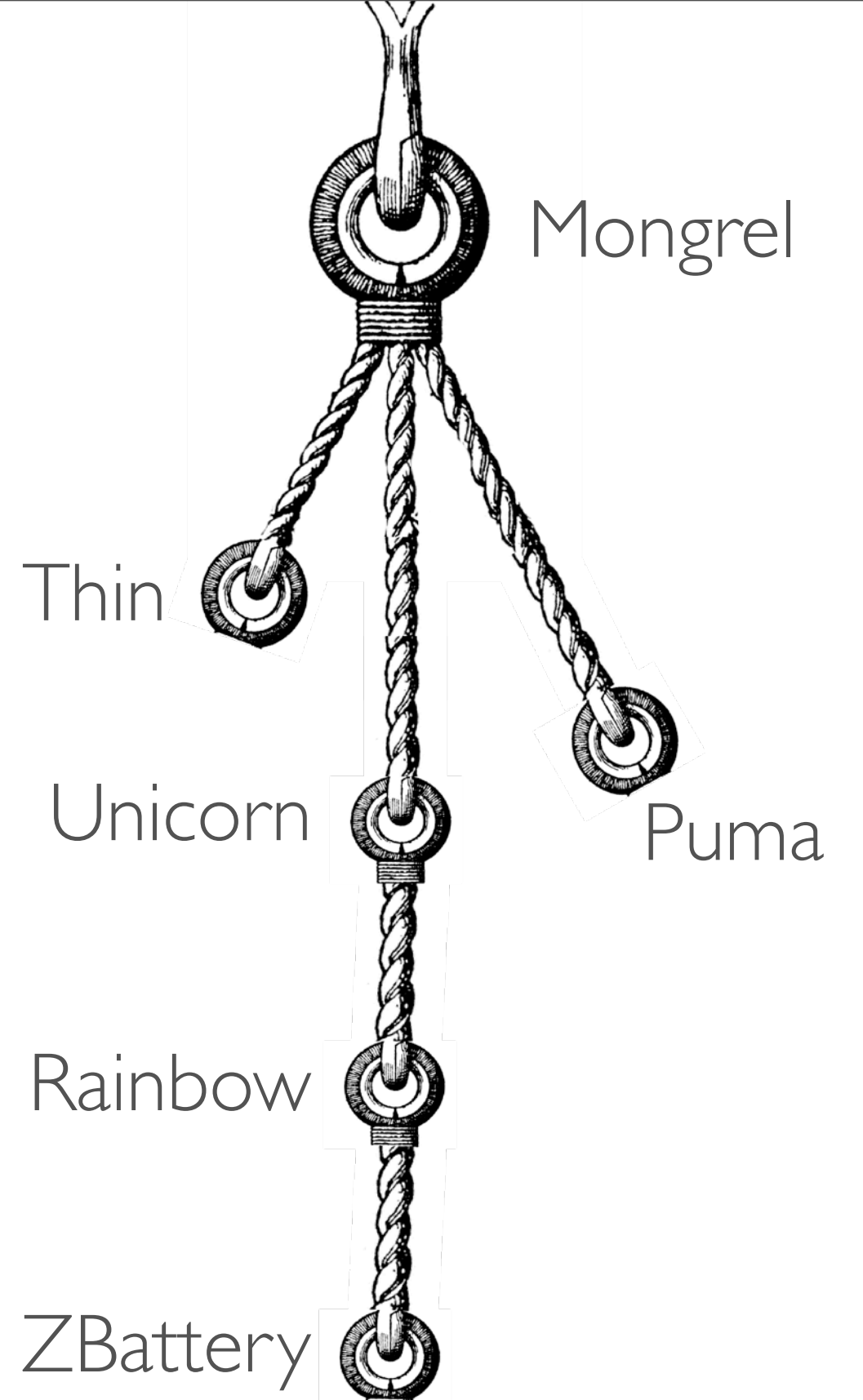
Passenger

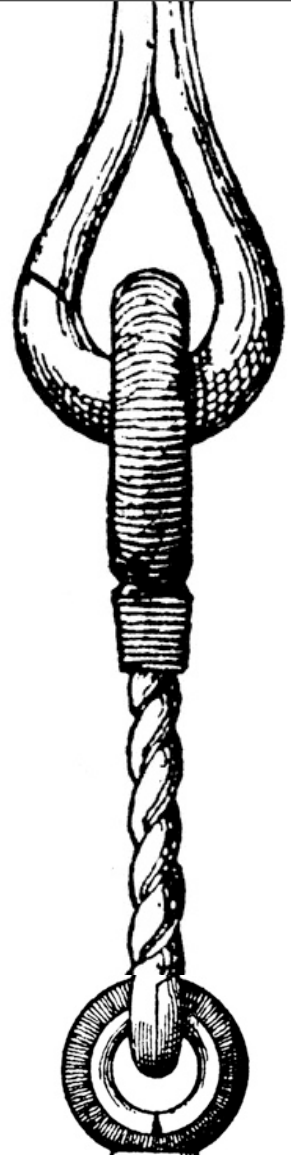


Yarn



Webrick

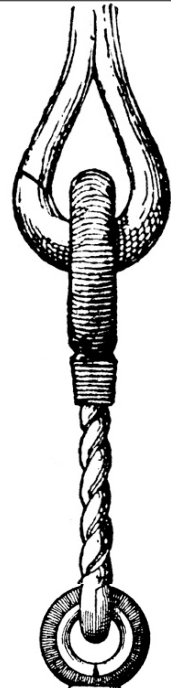




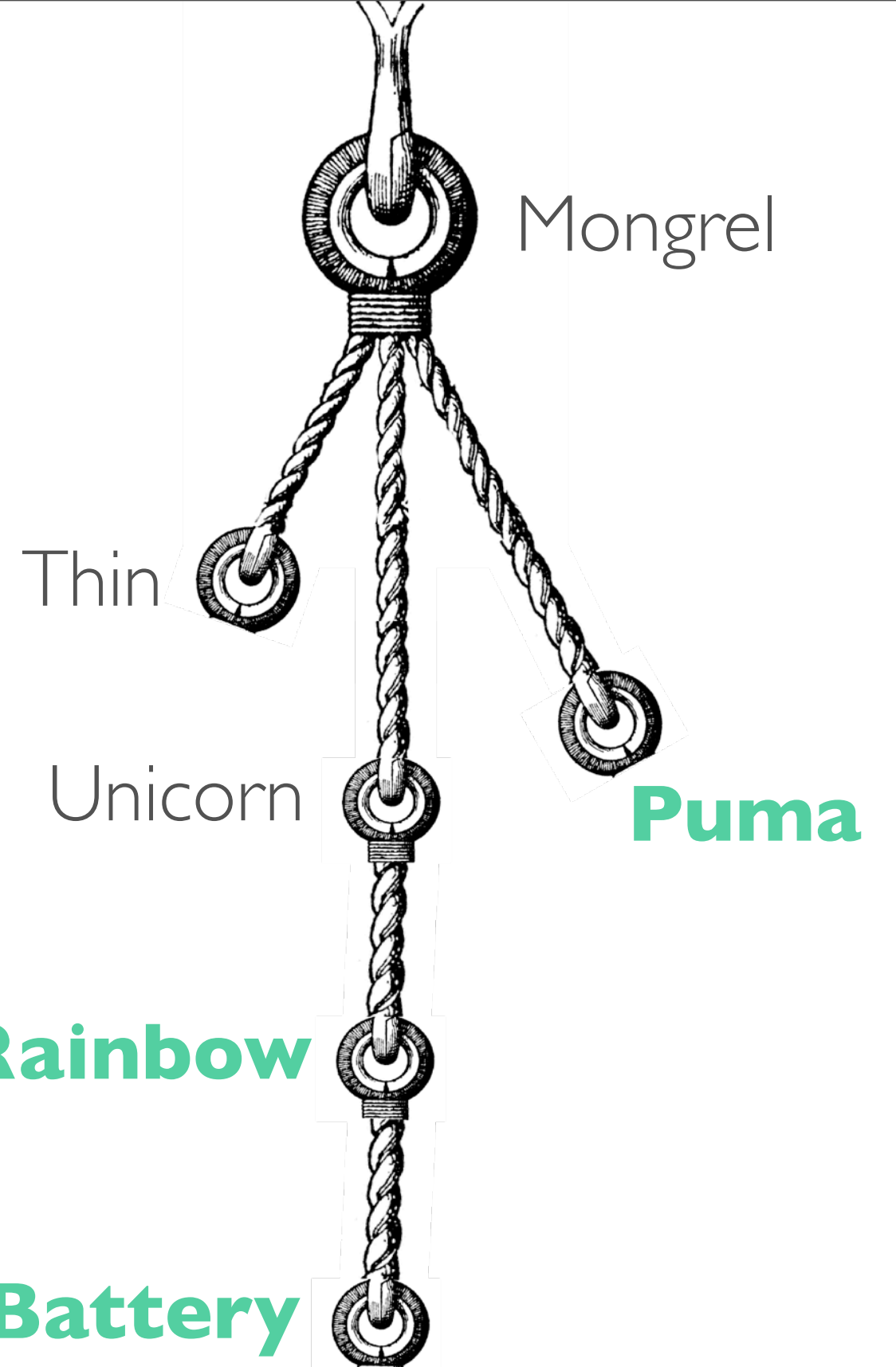
Passenger

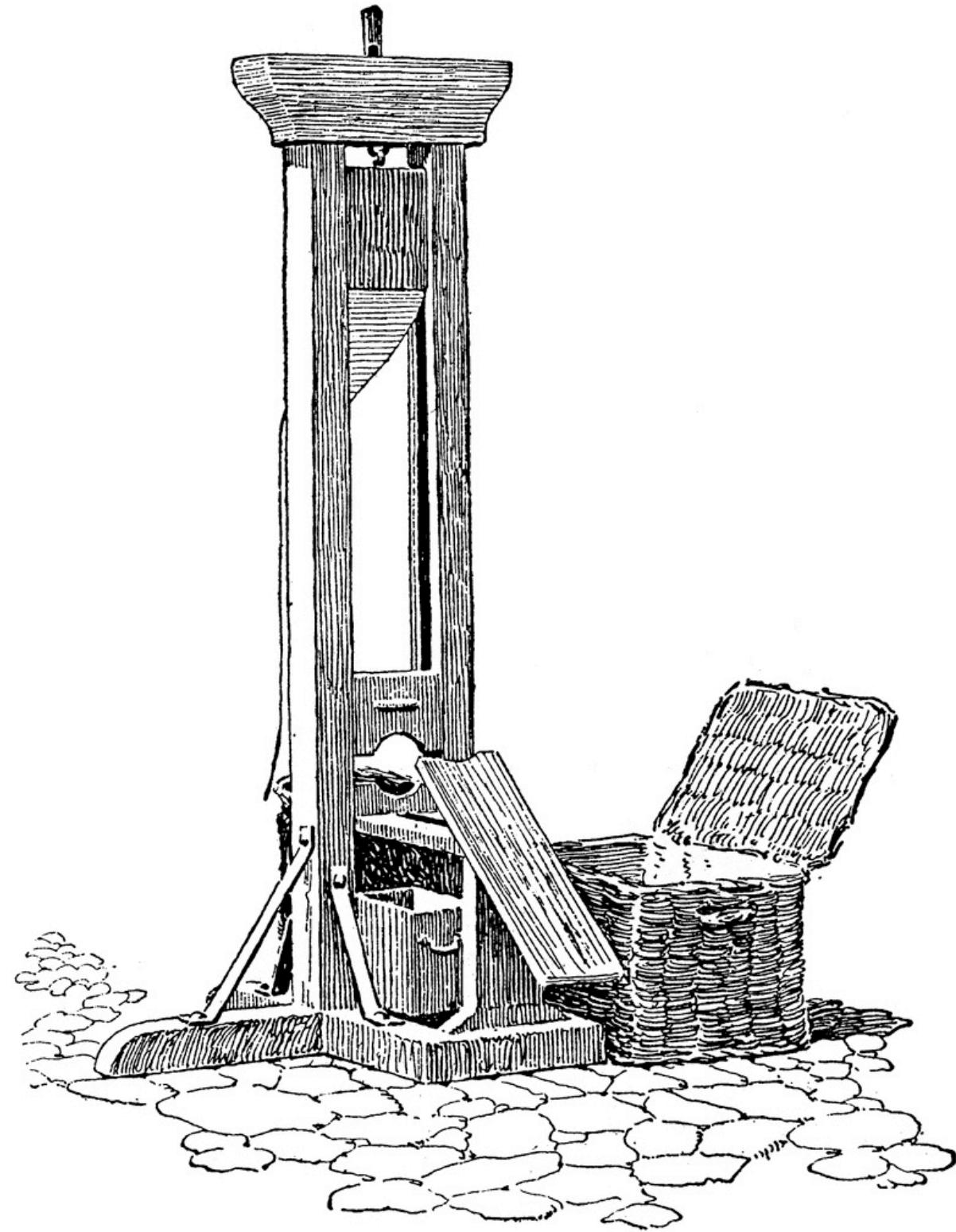


Yarn



Webrick





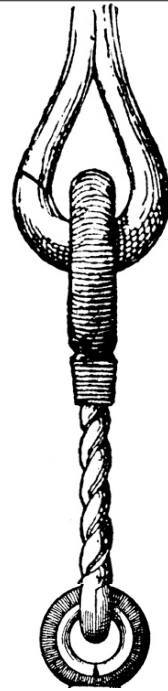
we had to kill the unicorns



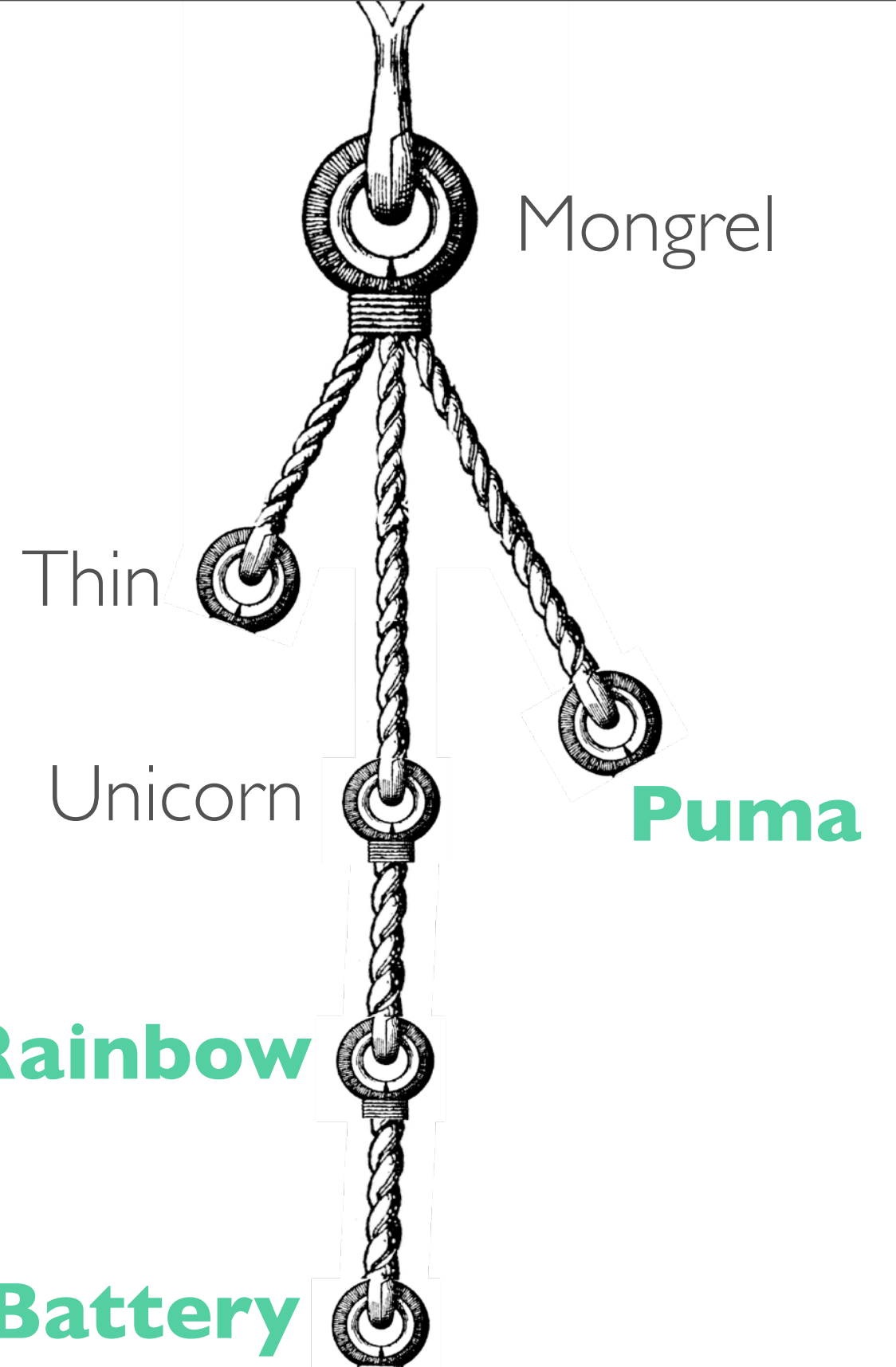
Passenger



Yarn



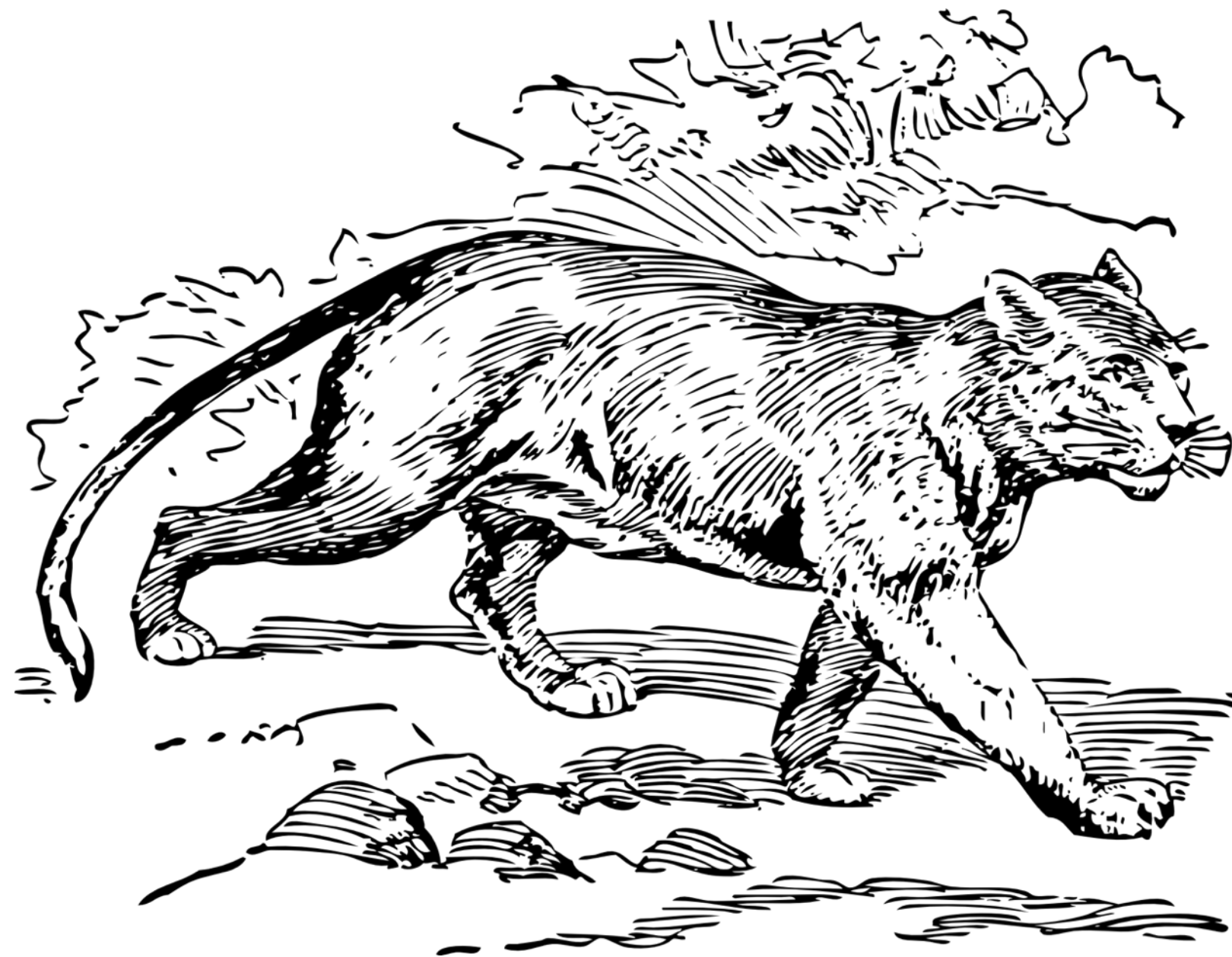
Webrick

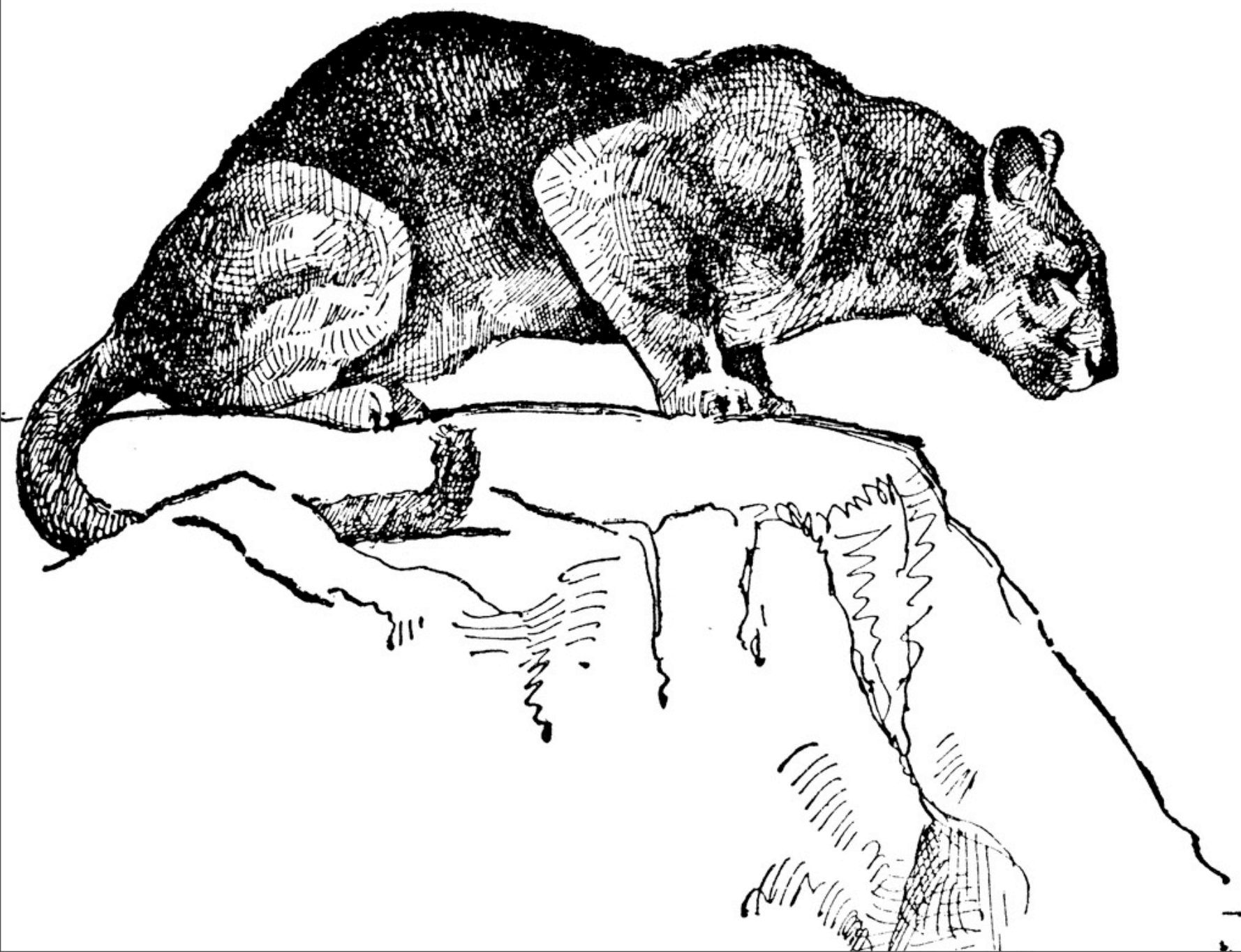




Puma

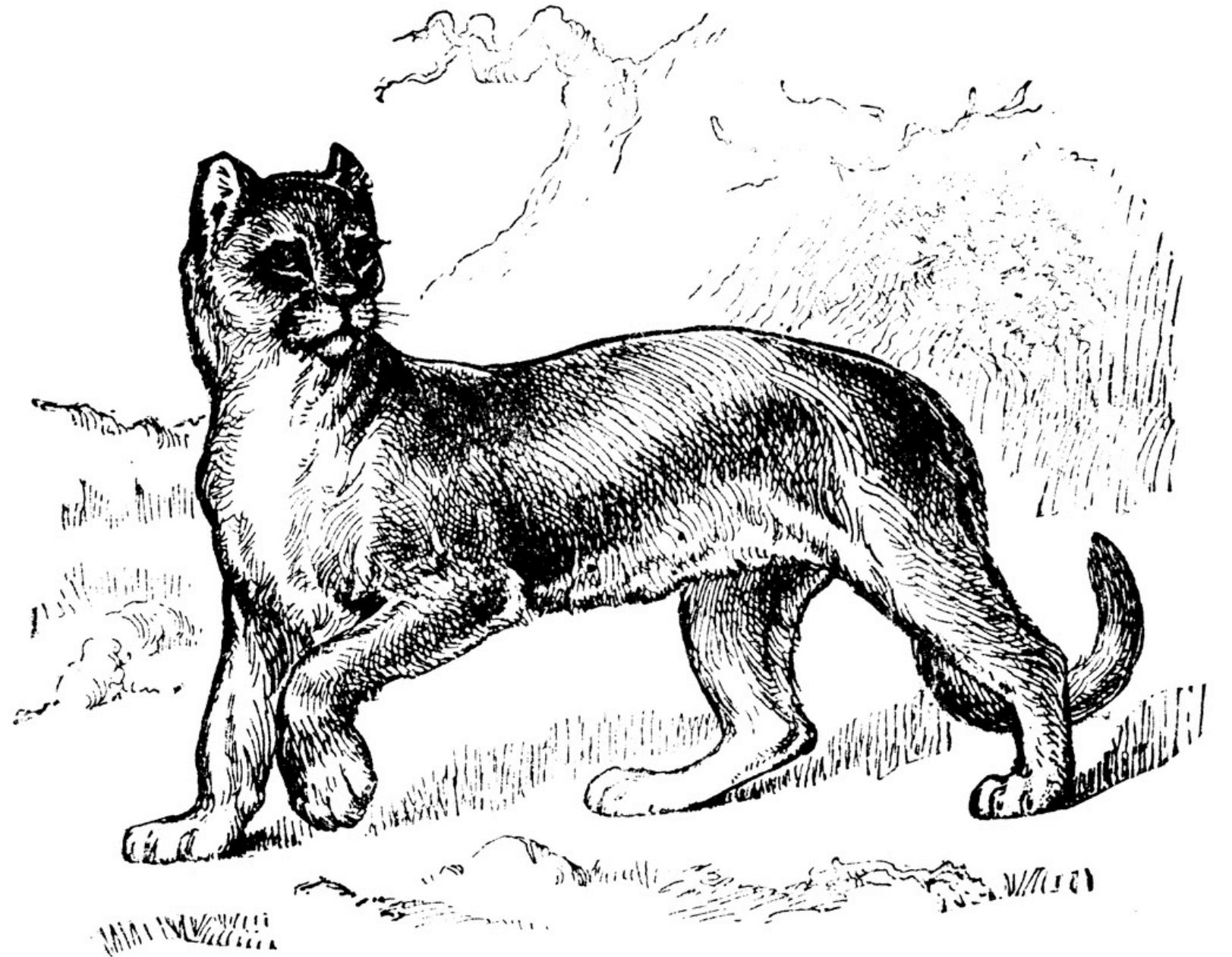
stable version vs development version

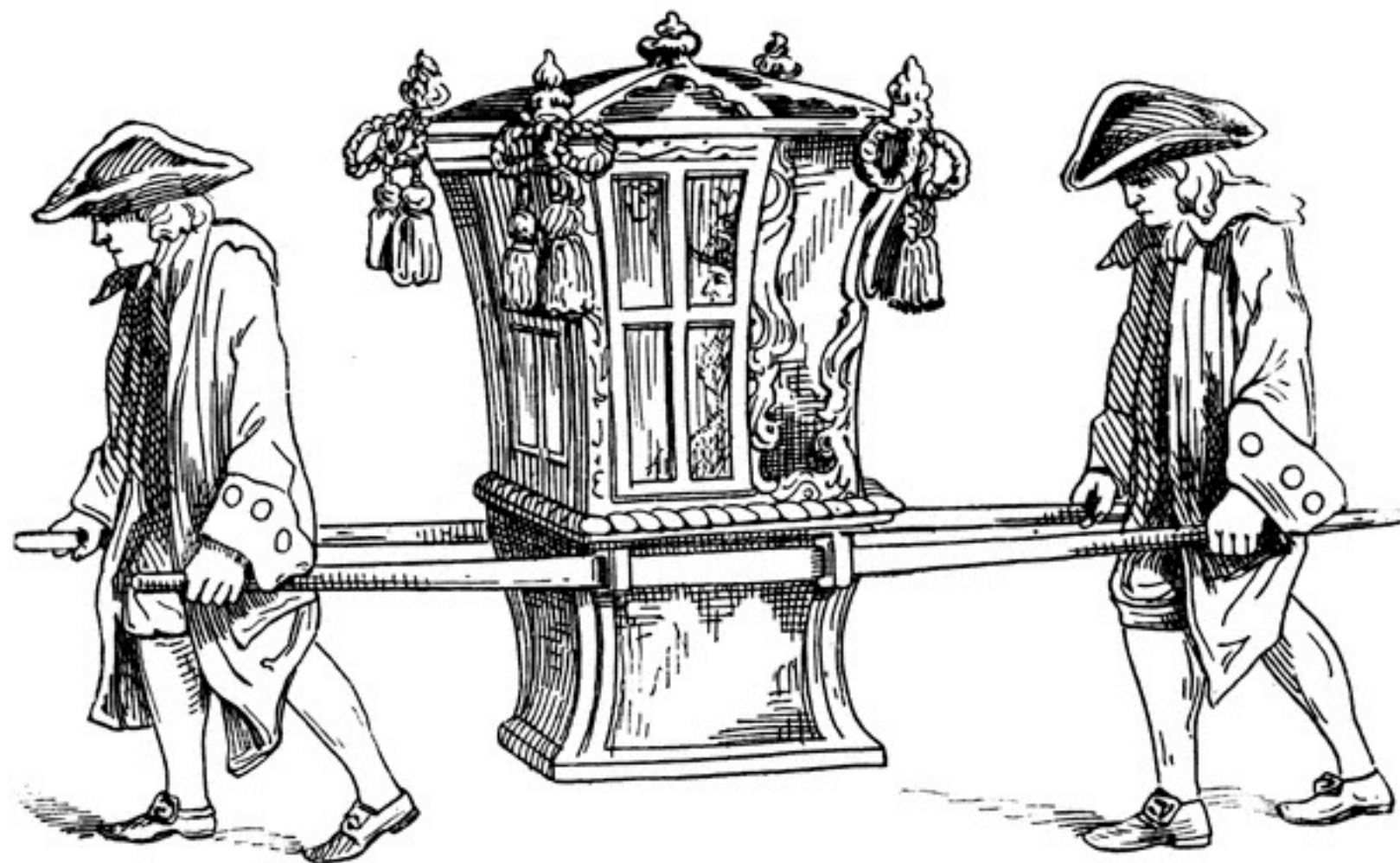




lack of rolling restarts

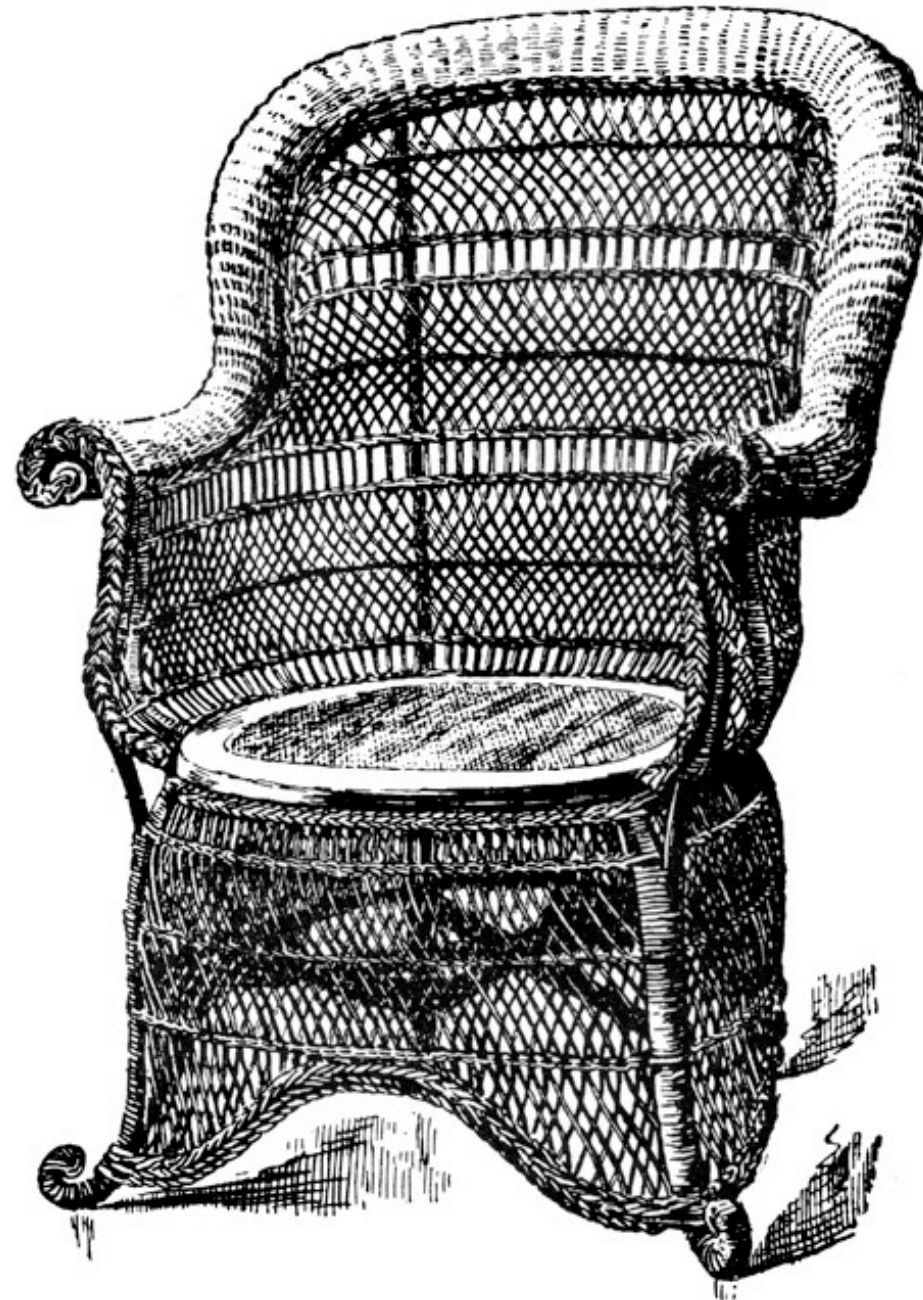
leaking memory

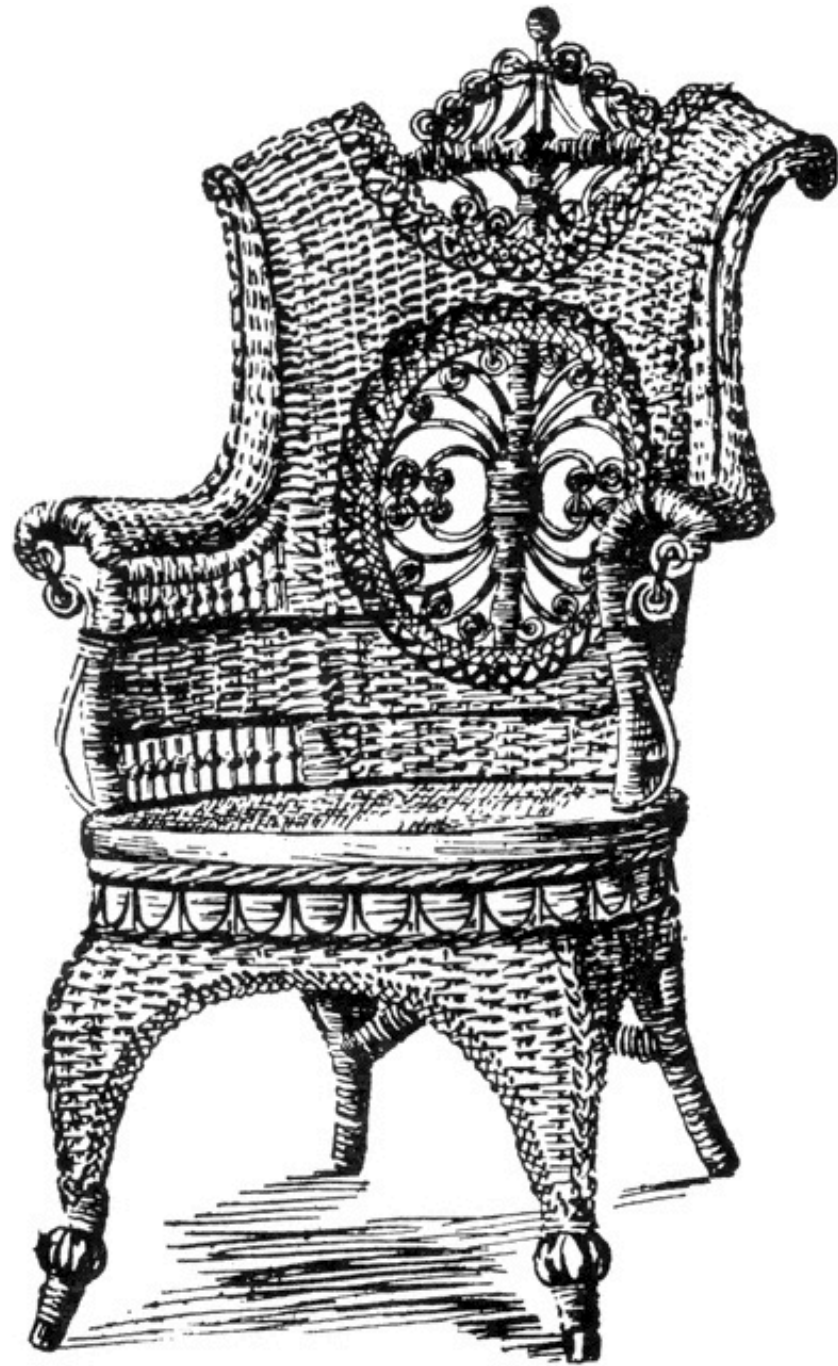




Passenger EE
4.0

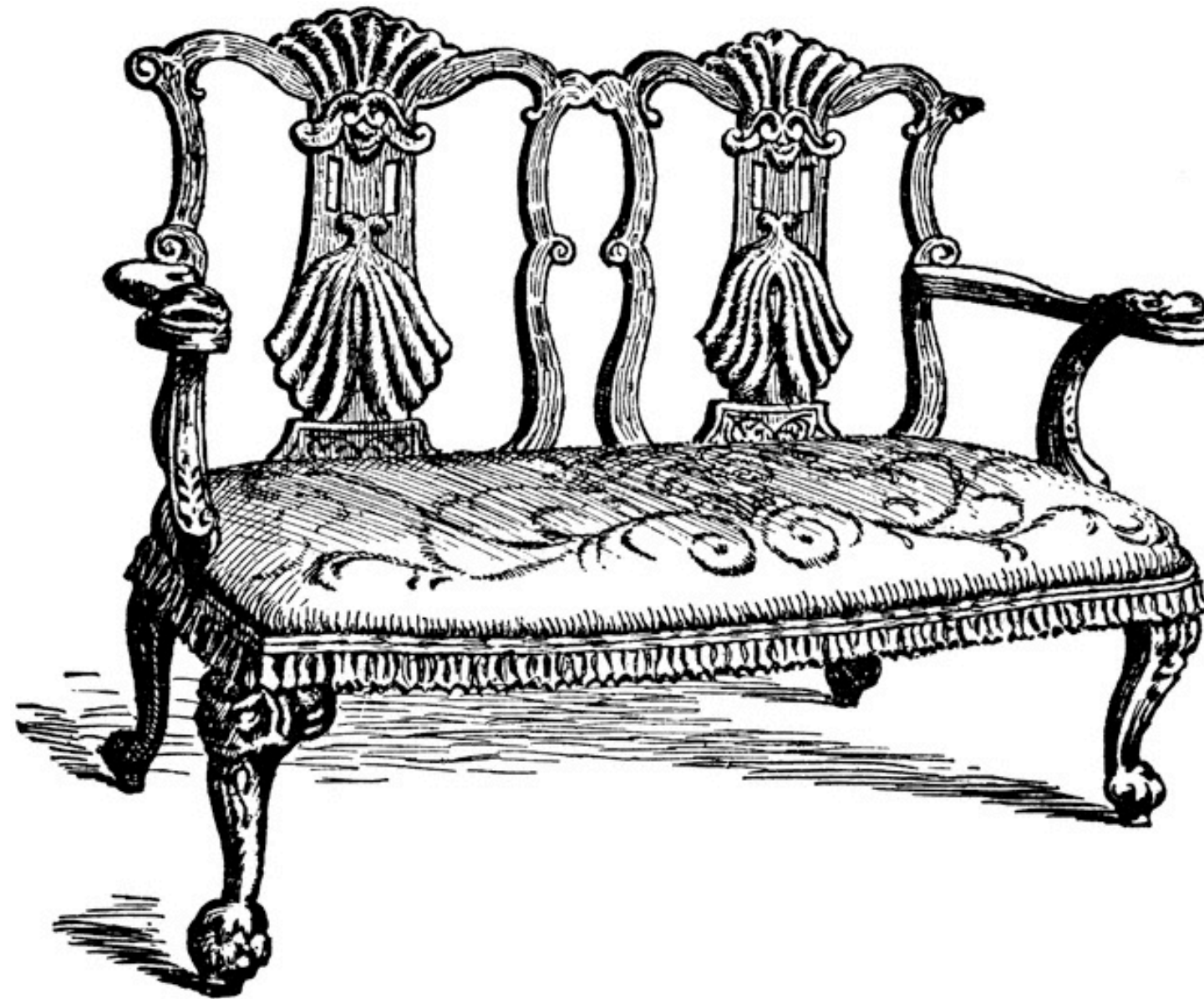
multi-threaded



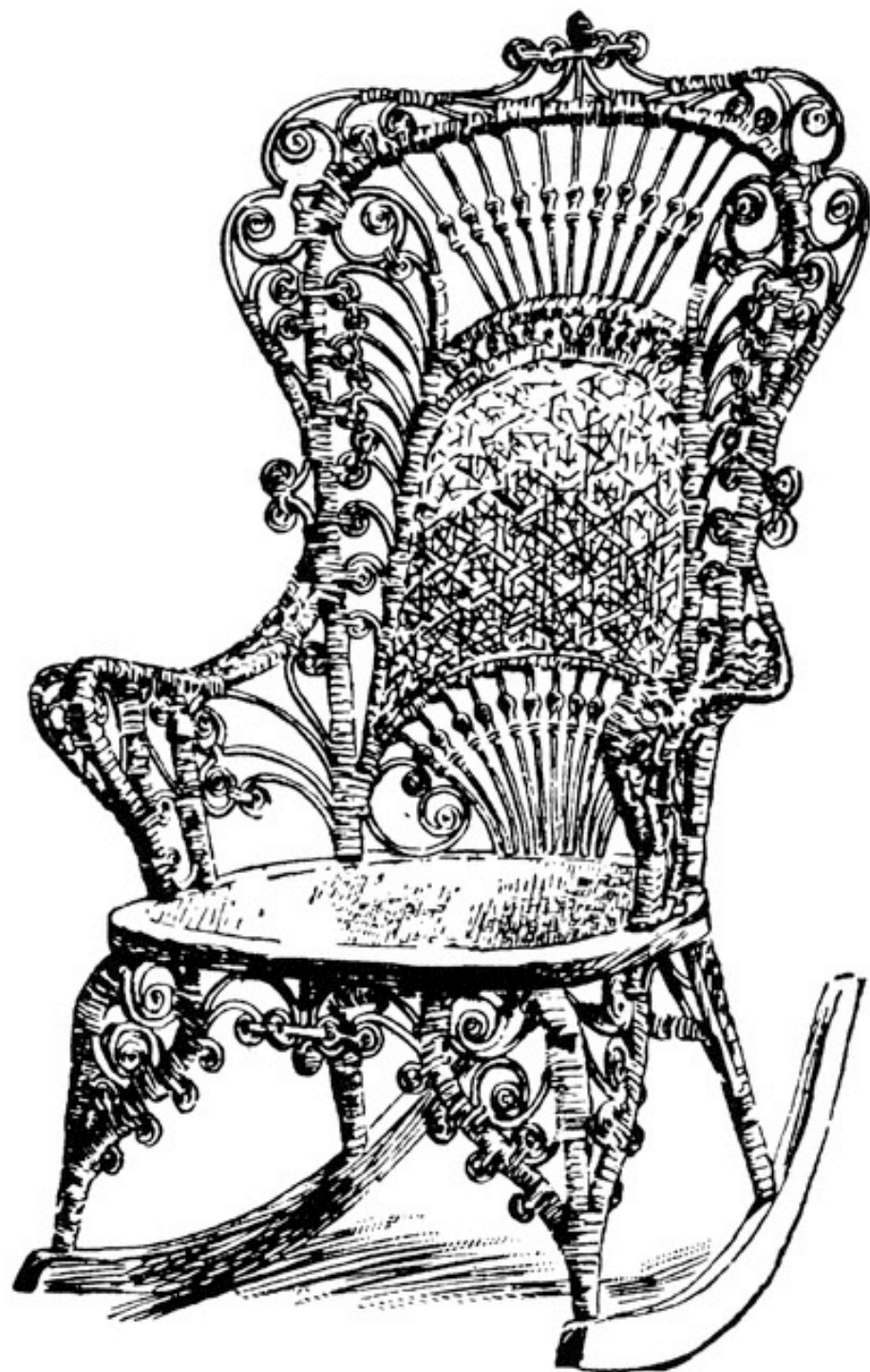


rolling restarts

fewer memory problems



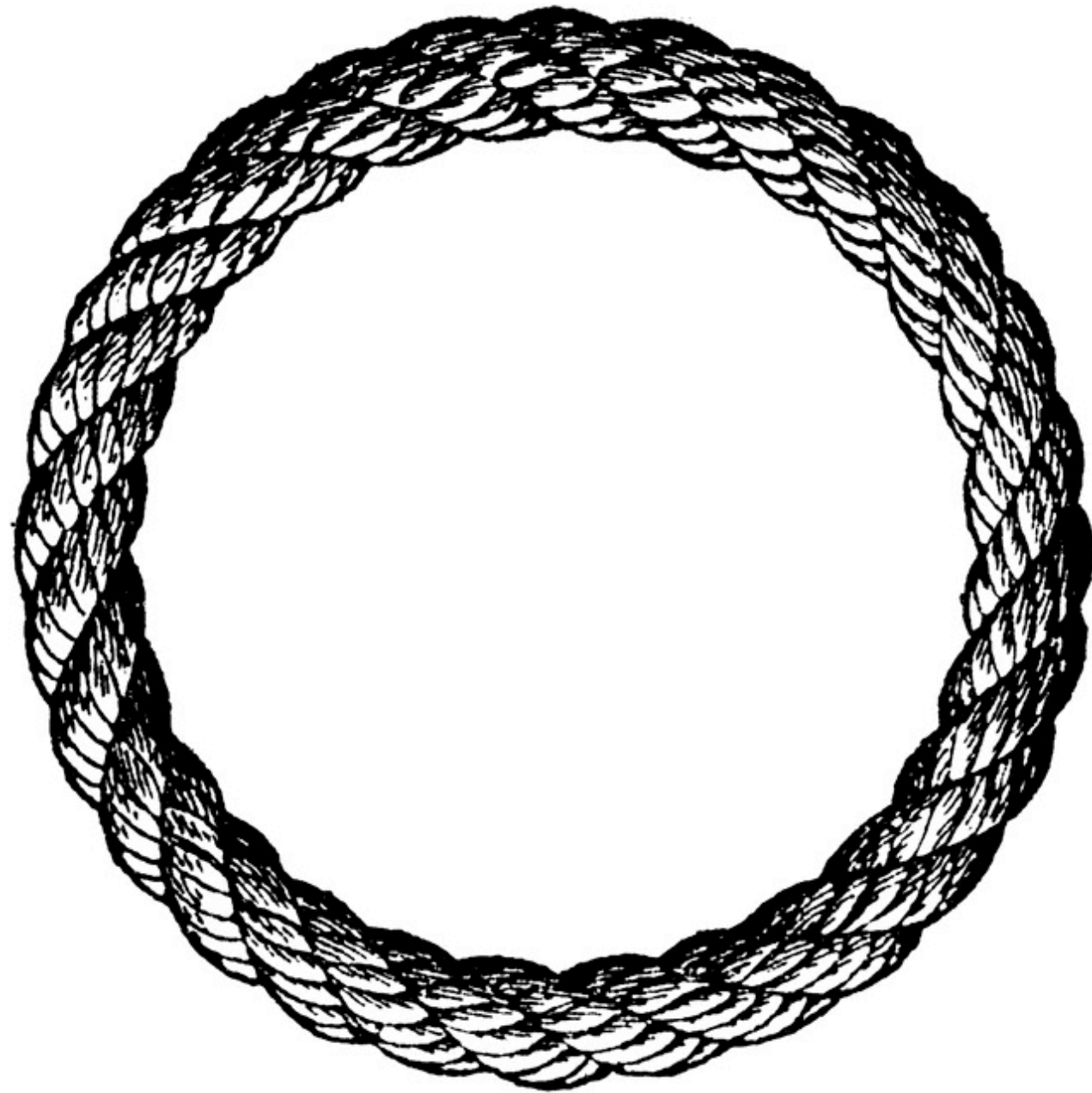
live IRB console



Problem 4

Threads



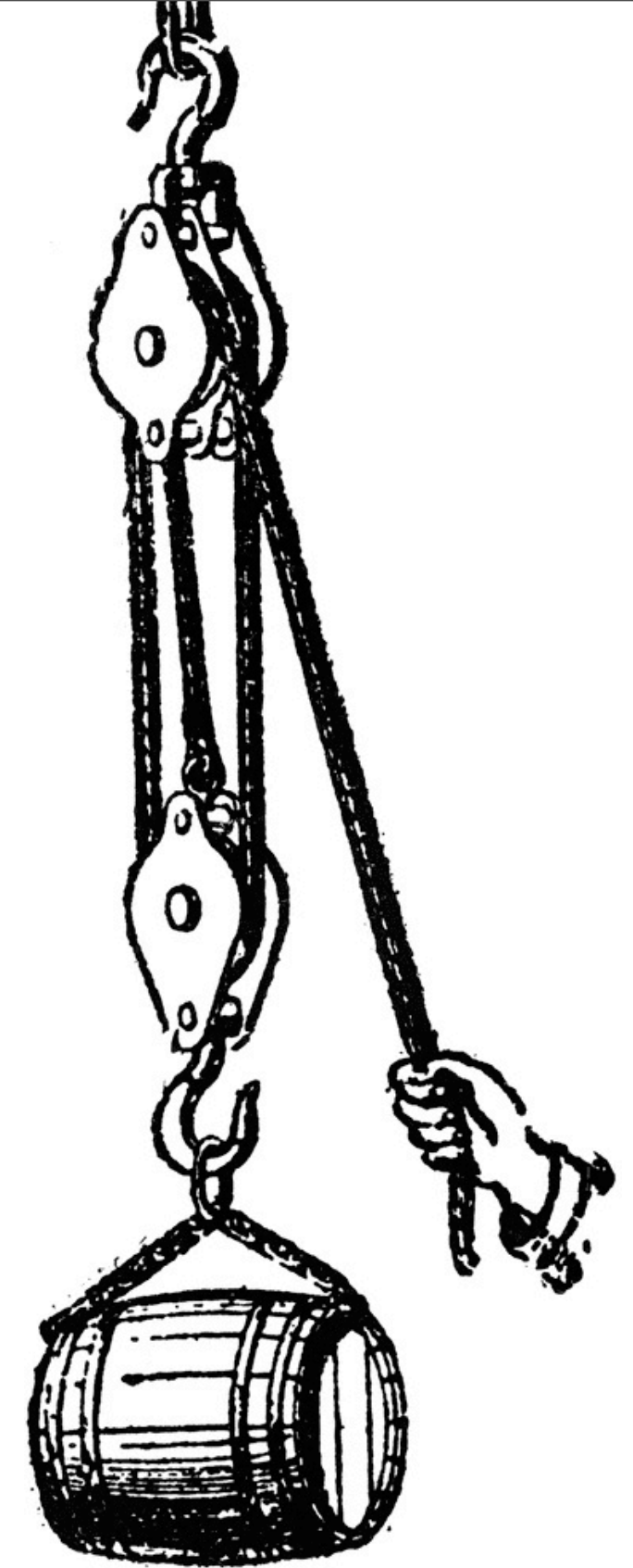


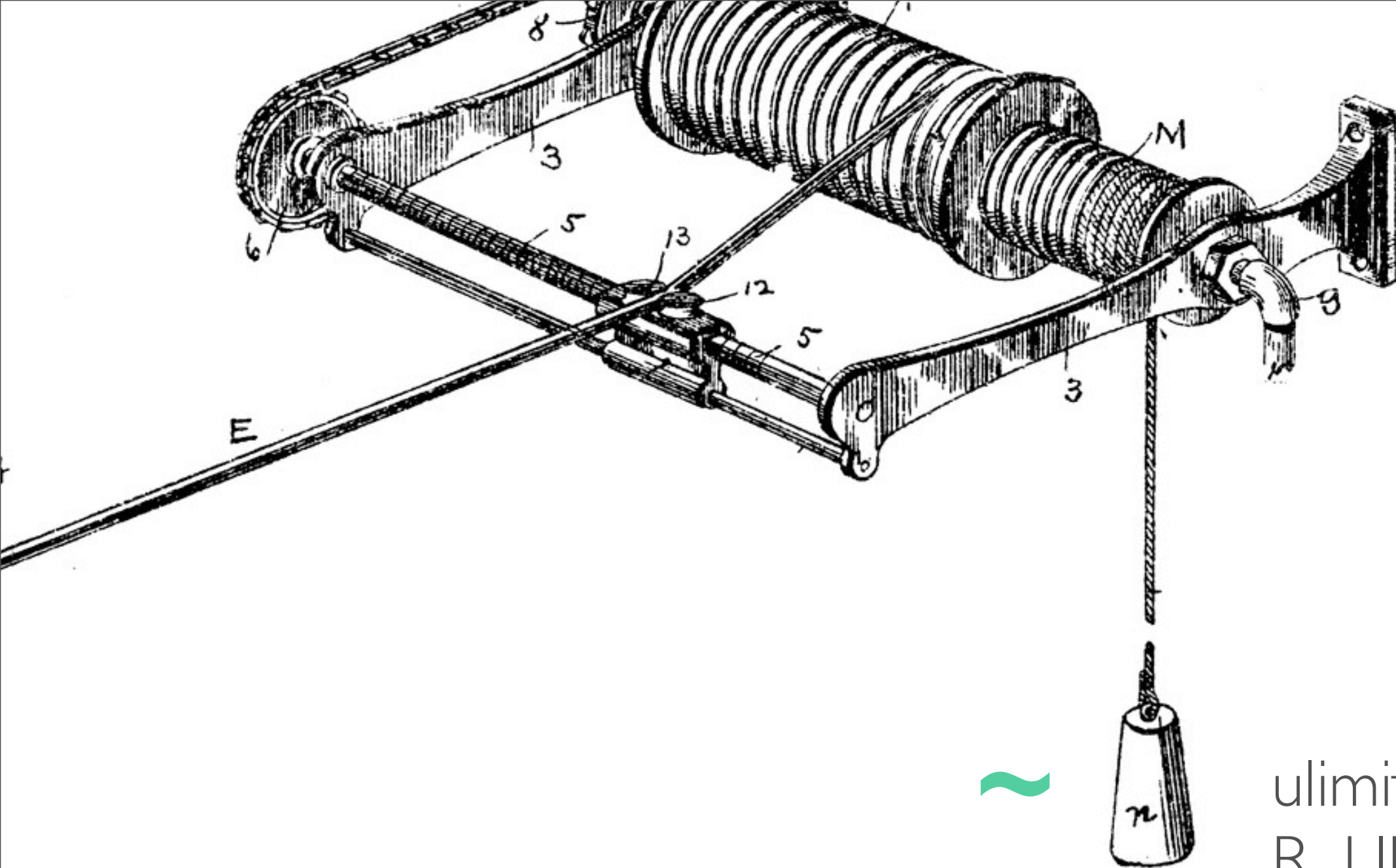
Ruby is thread-safe
Rails is thread-safe

what about your code?

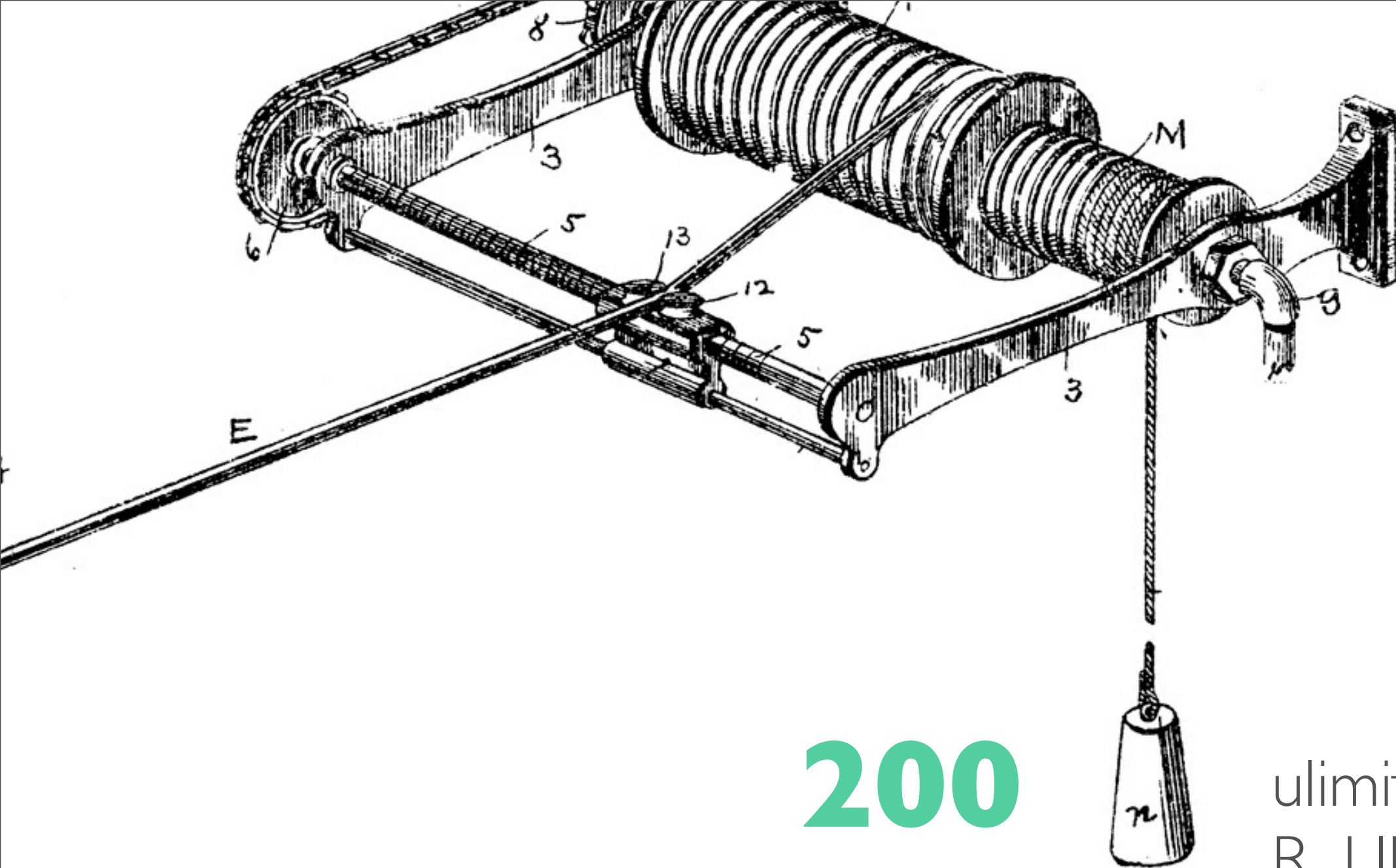


Thread.current[:var] is the new @@var



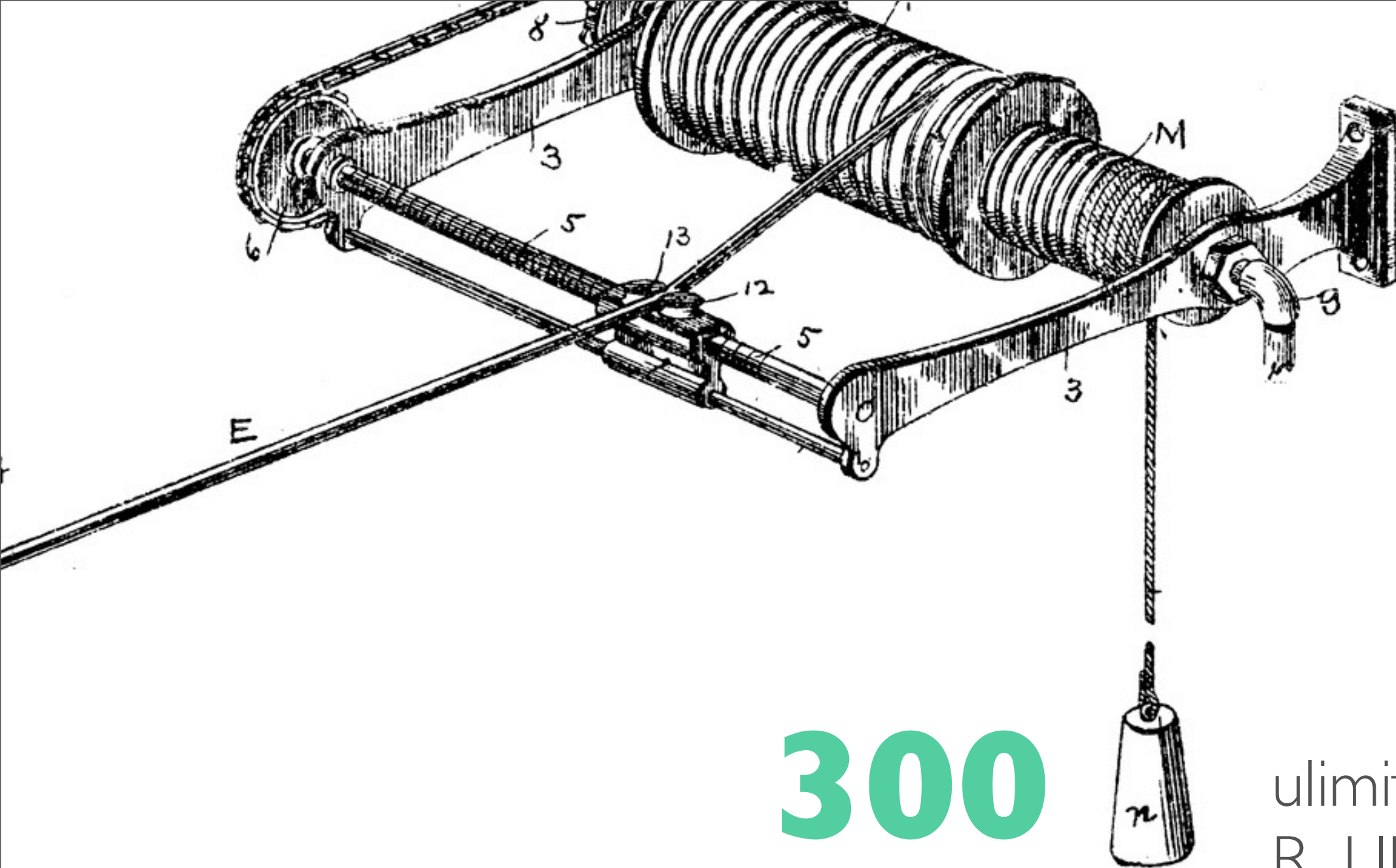


ulimit -n
R_LIMIT_NOFILE



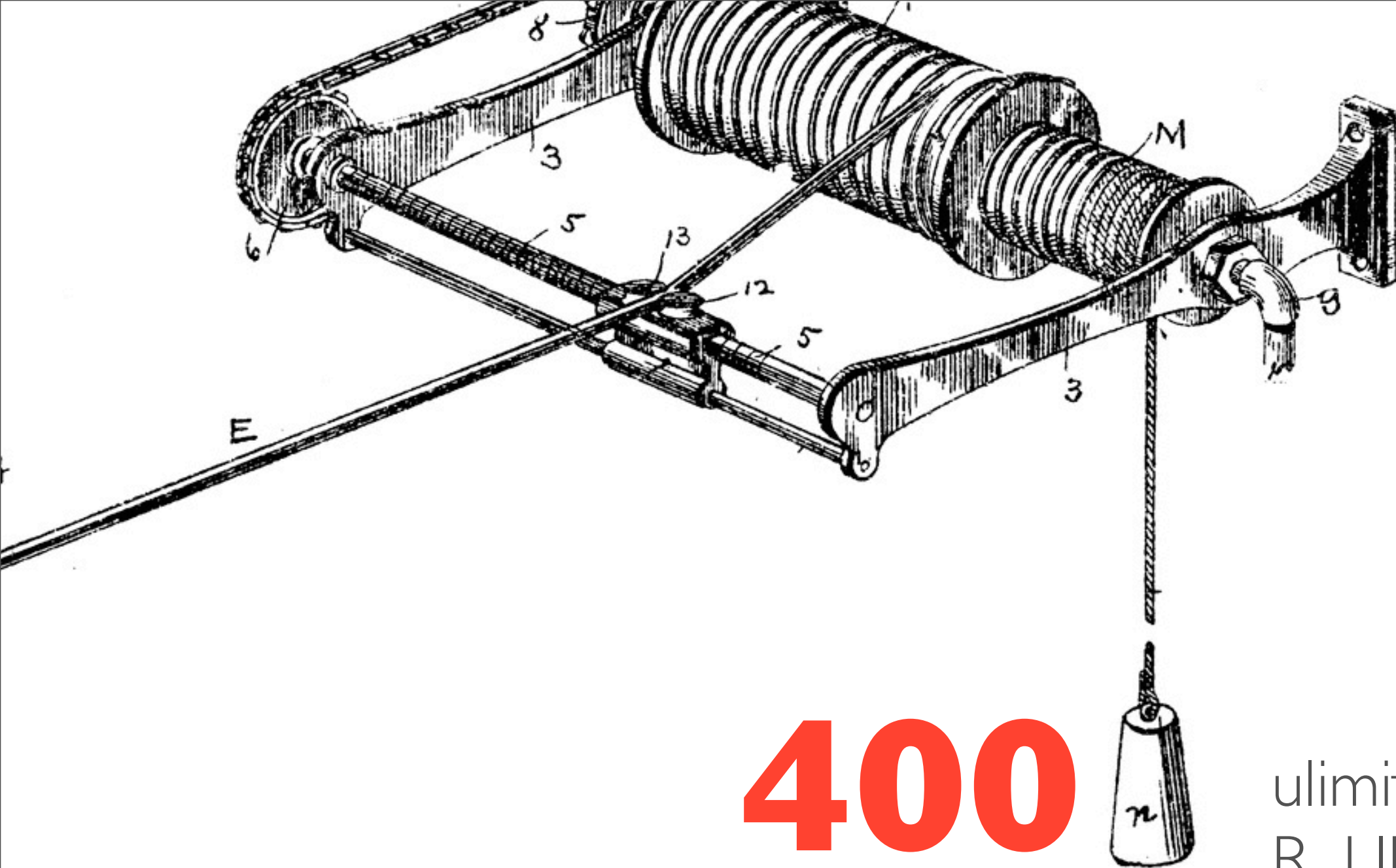
200

ulimit -n
R_LIMIT_NOFILE

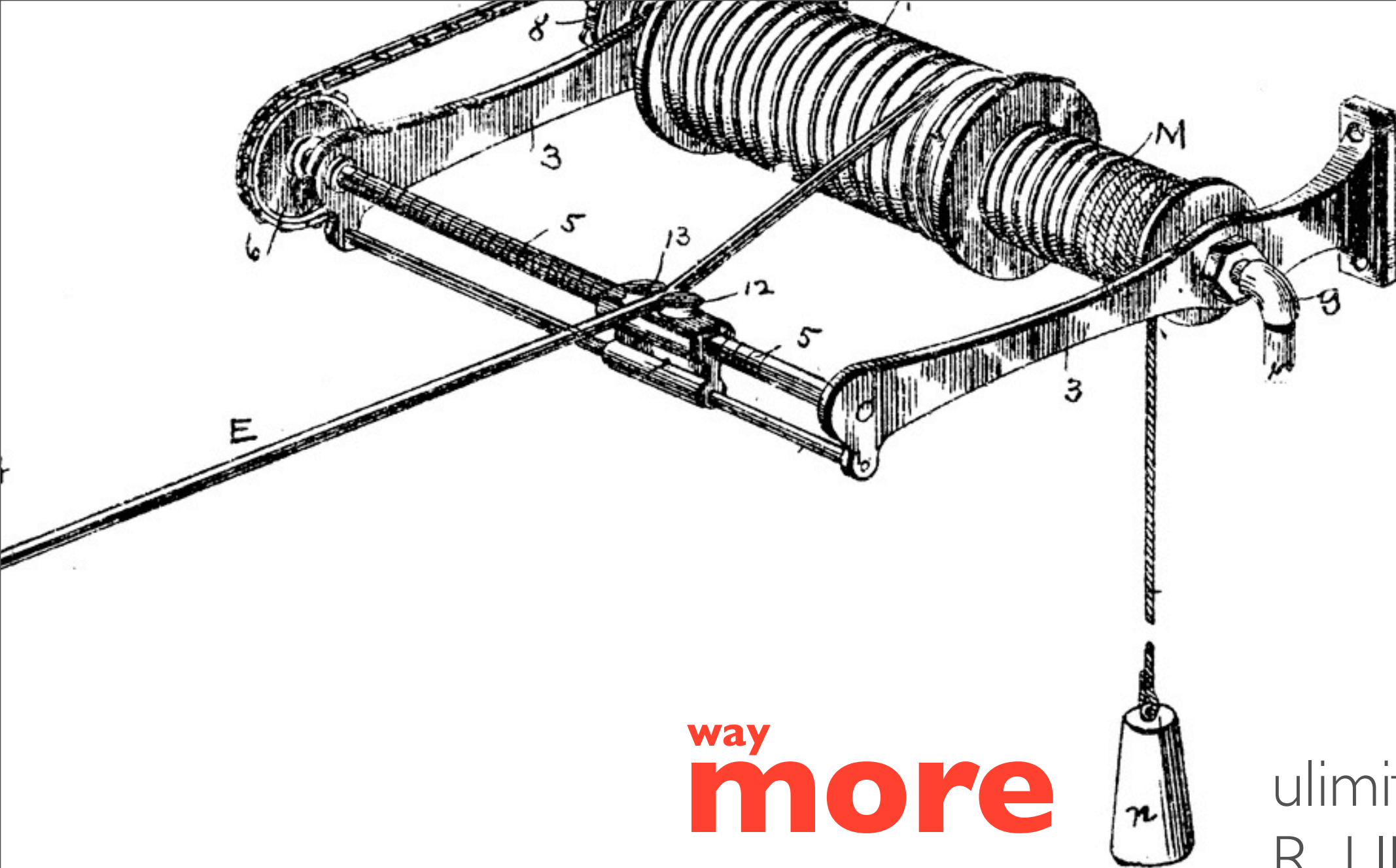


300

ulimit -n
R_LIMIT_NOFILE



ulimit -n
R_LIMIT_NOFILE

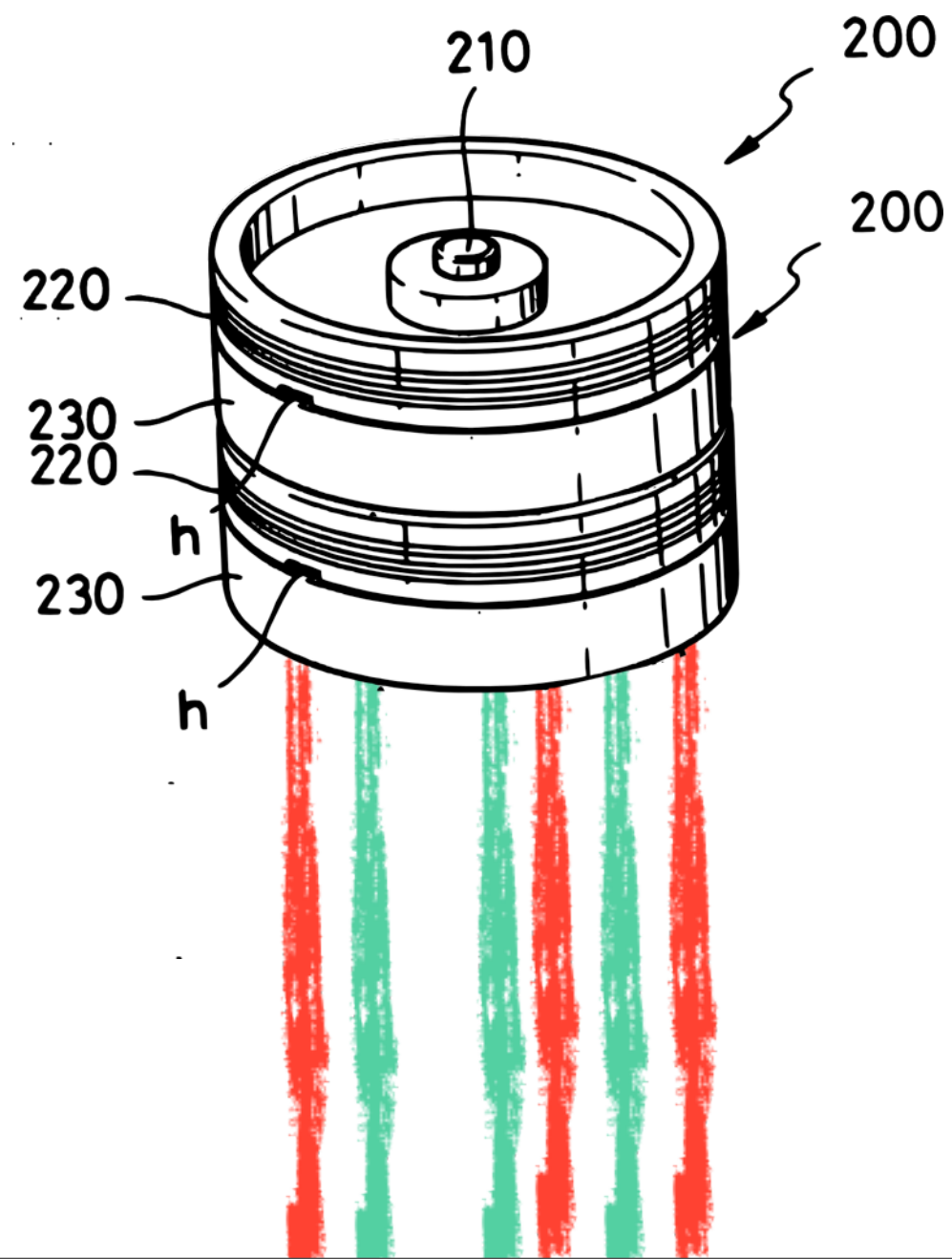


way
more

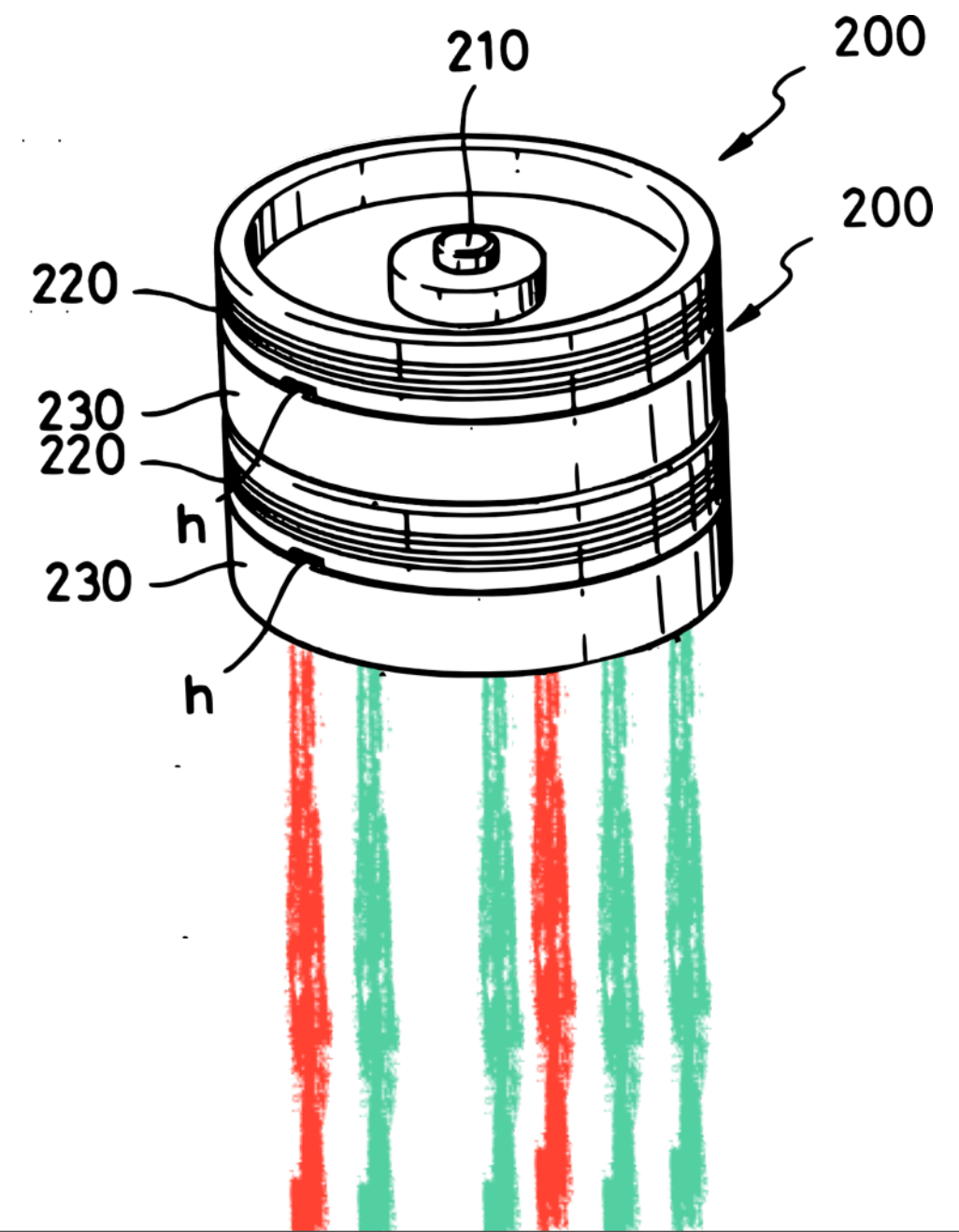
ulimit -n
R_LIMIT_NOFILE

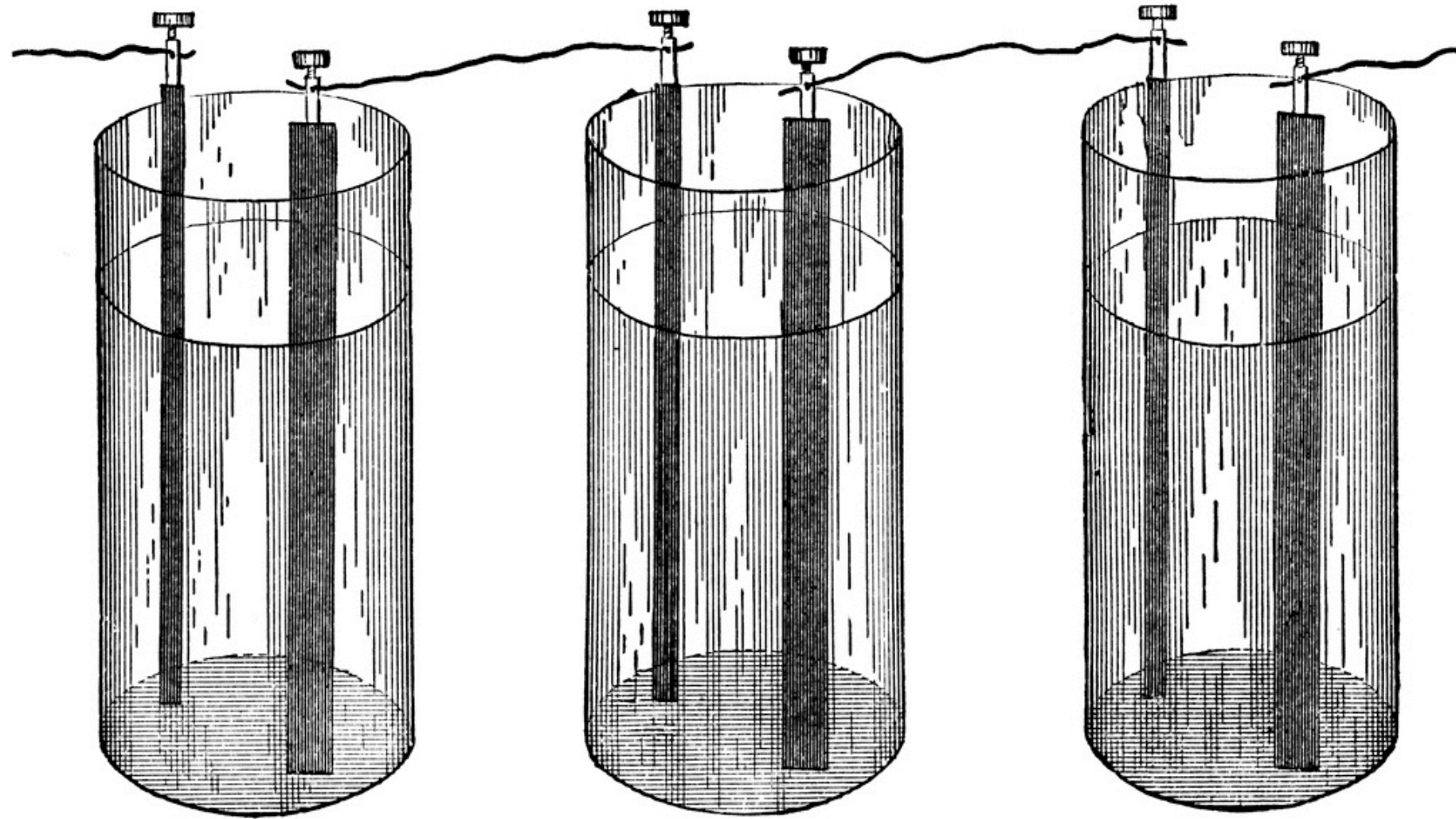
>65K

limit your DB connections

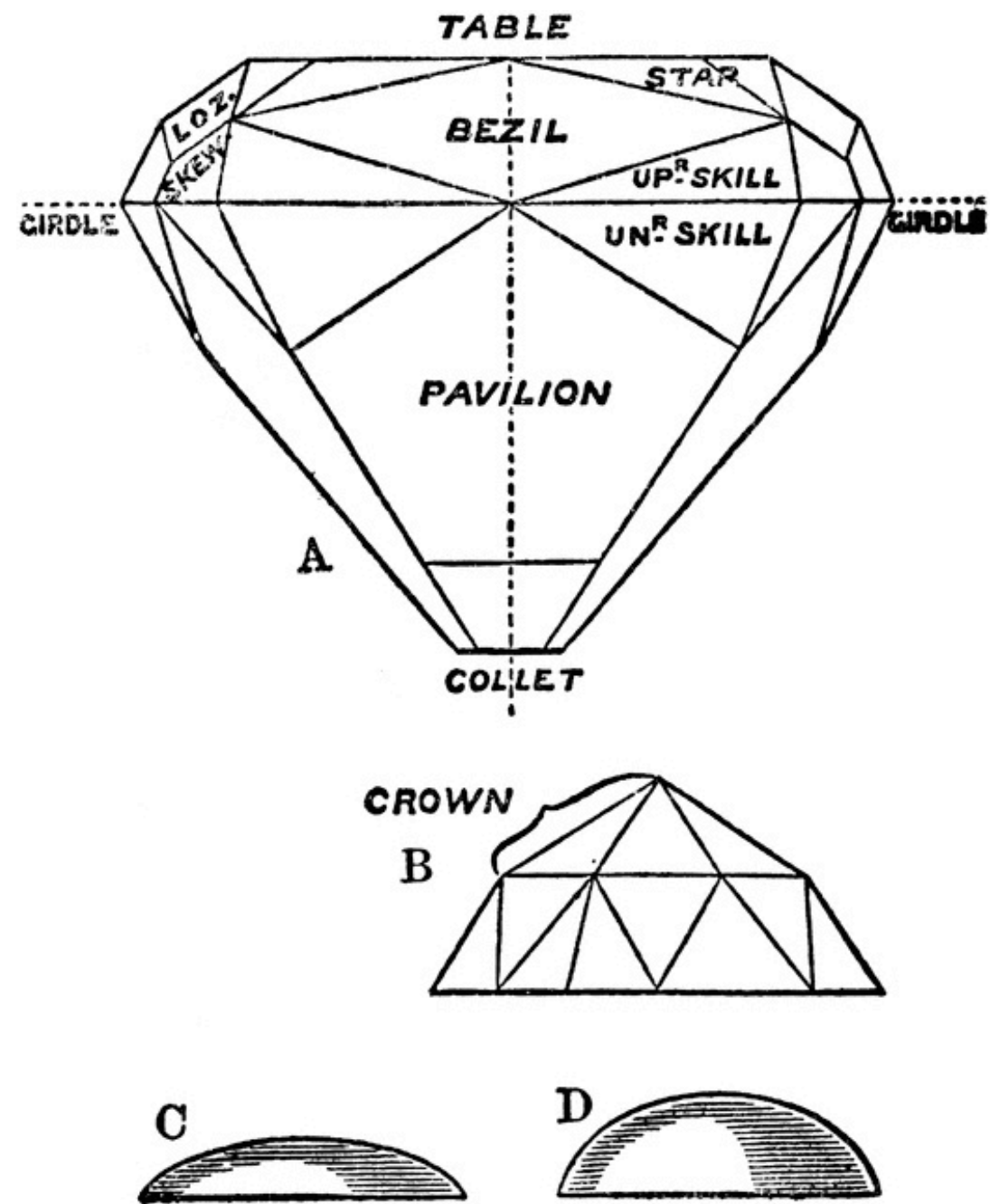


limit your cache server connections

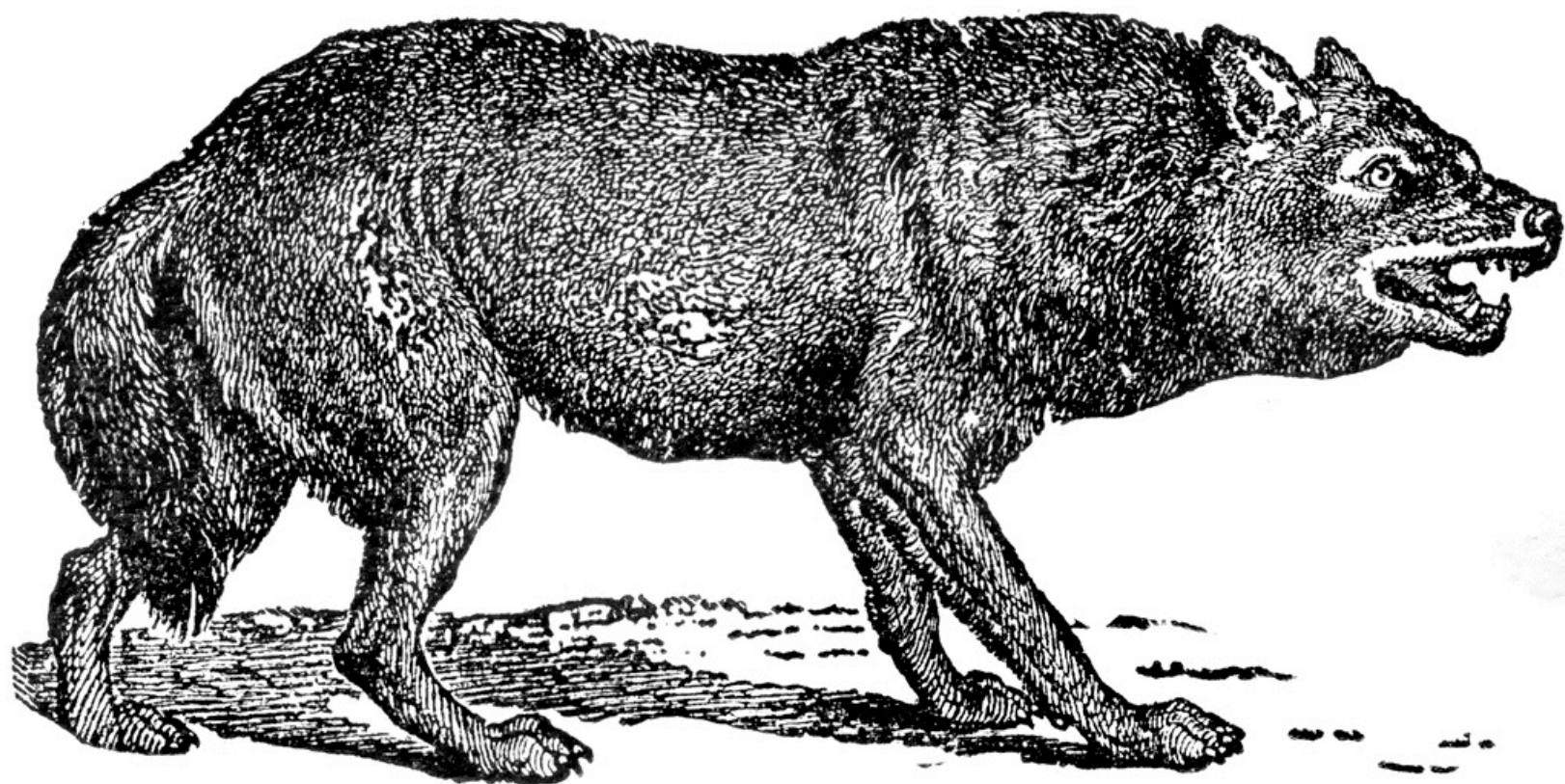




pool of connections



gem 'connection_pool'

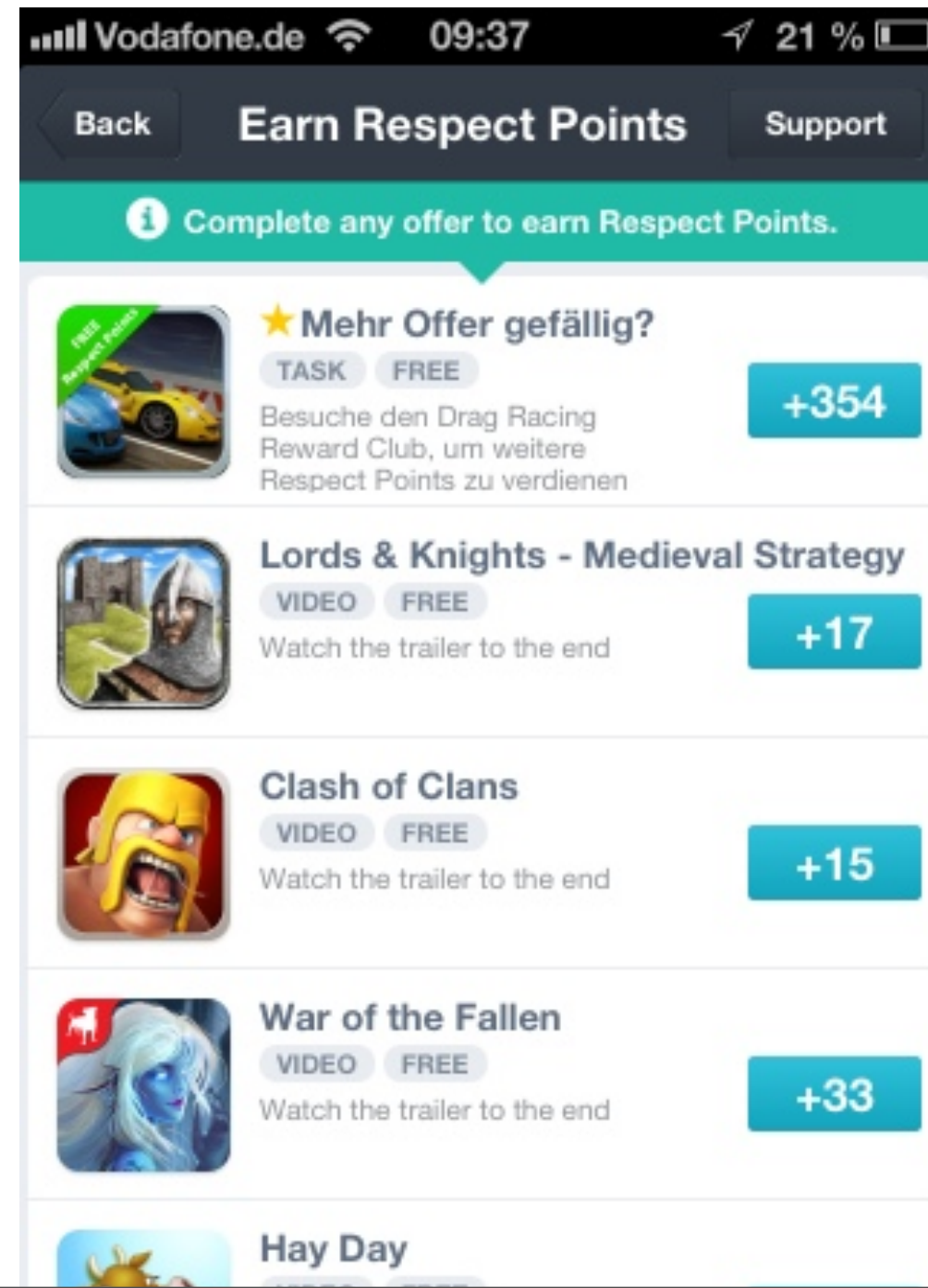


more problems?

we did it!



The App
now works in multi-threaded mode



don't be afraid of the monster!



Thanks



Q&A

Grzegorz Witek

@arnvald

Simon Kröger

@kroegerberlin

SponsorPay™

