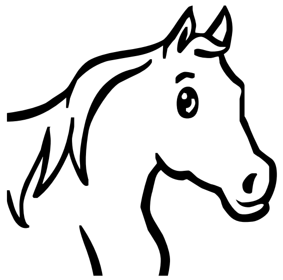
Computing vs Computers

The Houyhnhnm point of view



François-René Rideau, *TUNES Project*

LambdaConf 2016, 2016-05-28

http://ngnghm.github.io/talks/cc-lc2016.pdf

This Talk

Based on my blog *Houyhnhnm Computing*

`http://ngnghm.github.io/`

Itself based on my old *TUNES Project*

`http://TUNES.org/`

The Take Home Points

Interactions around us, not artifact below us

Orthogonal Persistence

Reflective Systems: First-class Computation

A change in point of view changes architecture

Plan of the Talk

Houyhnhnms vs Yahoos

Orthogonal Persistence

Reflective Architecture

The way forward

I. Houyhnhnms vs Yahoos

An Installation Story

Installing **`overtone`** on N’s Windows laptop

An Installation Story

Installing **`overtone`** on N’s Windows laptop

Clojure, Emacs, SuperCollider, JVM, Windows

5 platforms!

An Installation Story

Installing **`overtone`** on N’s Windows laptop

Clojure, Emacs, SuperCollider, JVM, Windows

5 platforms!

Issue: hunting files online

Issue: **`HOME`**: **`USERPROFILE`** or **`LOCALAPPDATA`**?

Issue: 32-bit SuperCollider vs 64-bit JVM

An Installation Story

Installing **`overtone`** on N's Windows laptop

Clojure, Emacs, SuperCollider, JVM, Windows

5 platforms!

Issue: hunting files online

Issue: **`HOME`**: **`USERPROFILE`** or **`LOCALAPPDATA`**?

Issue: 32-bit SuperCollider vs 64-bit JVM

Linux (Ubuntu): I had made it worked earlier

Issues: manual config, **`pulseaudio`** vs **`jackd`**

No *solution* — but *scriptable workaround*

N wonders

How do you trust random programs?

How can programs disagree on homedir? On ABI?

How is the discrepancy not at least detected?

Why configure at all? Why no sane defaults?

What are files?

Why redo everything on every machine?

Why five development platforms?

Why can’t it all be automated?

Why is the user involved at all?

N is ignorant

N is obviously ignorant

But not *stupid*

N doesn’t know how Humans compute

Yet N is an computing expert

N is ignorant

N is obviously ignorant

But not *stupid*

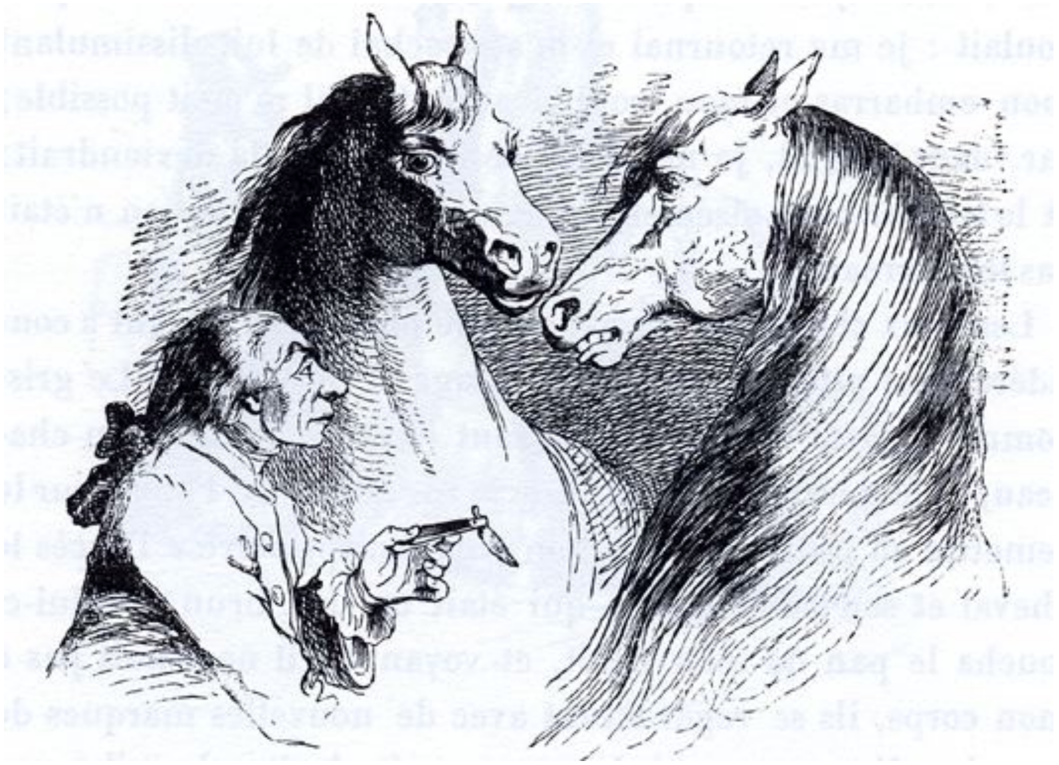
N doesn’t know how Humans compute

Yet N is an computing expert

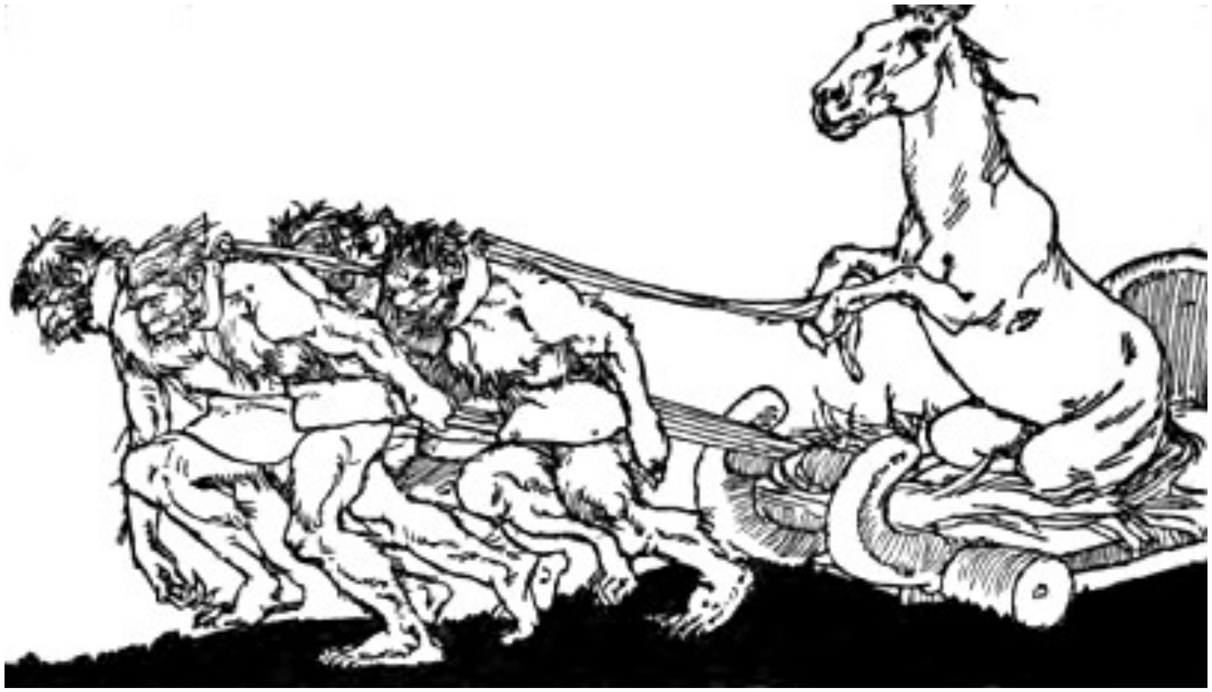
N is not Human, but a Houyhnhnm

Gulliver

Gulliver and Houyhnhnms

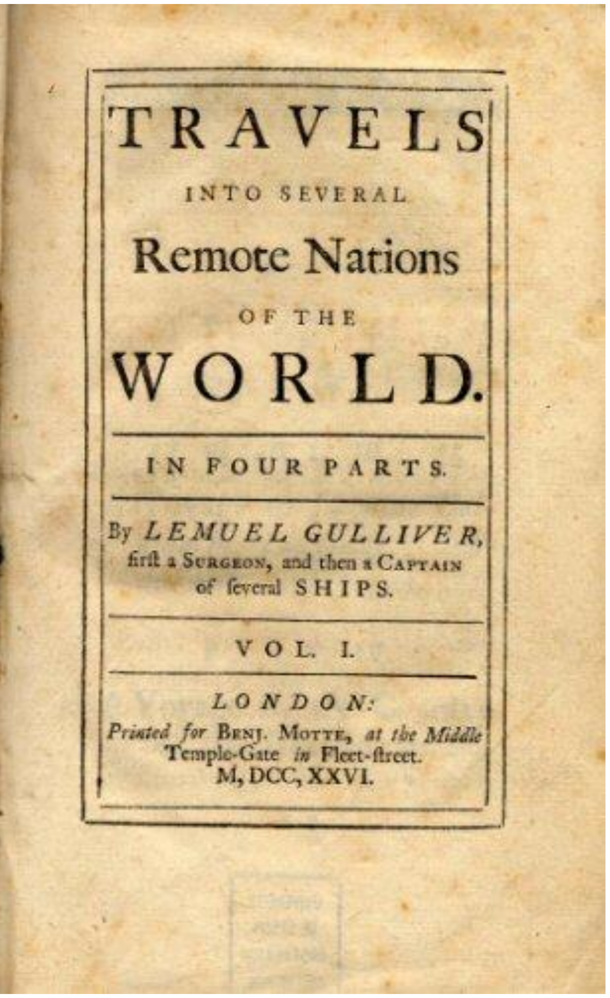


Houyhnhnms and Yahoos



Swift's Book

CAPTAIN LEMUEL GULLIVER OF REDRIFF ÆTAT. SUÆ LVIII.
Compoſitum jus, faſque animi, ſanctoſque receſſus
Mentis, et incoctum generoſo pectus honeſto.


TRAVELS
INTO SEVERAL
Remote Nations
OF THE
WORLD.

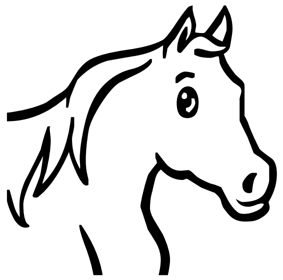
IN FOUR PARTS.

By *LEMUEL GULLIVER*,
firſt a SURGEON, and then a CAPTAIN
of ſeveral SHIPS.

VOL. I.

LONDON:
Printed for BENJ. MOTTE, *at the Middle*
Temple-Gate *in* Fleet-ſtreet.
M, DCC, XXVI.

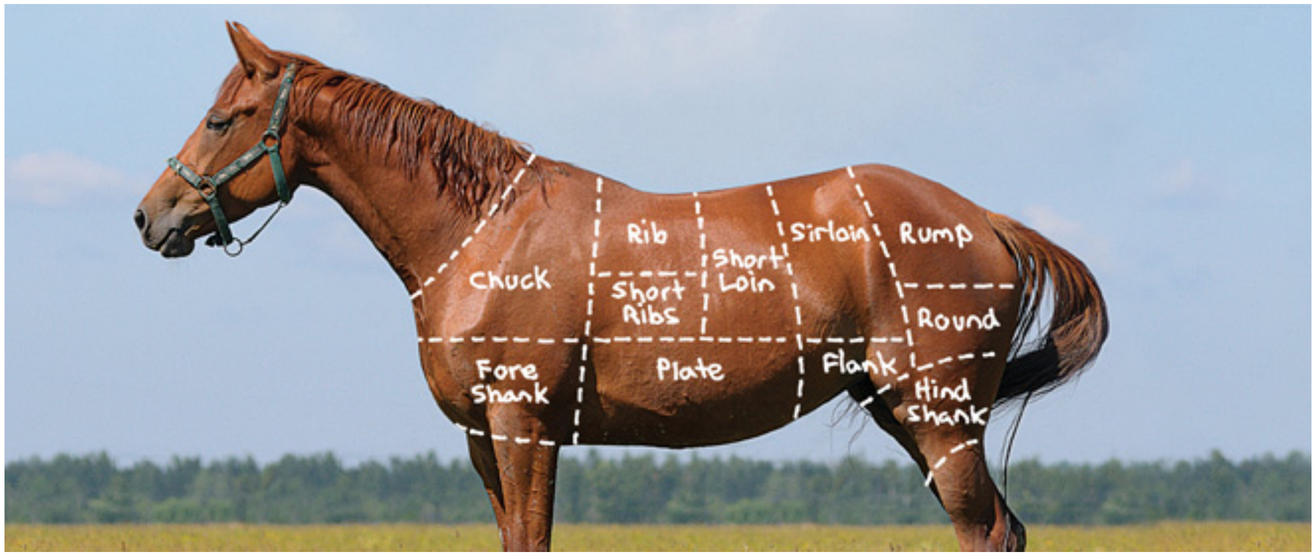
Ngnghm



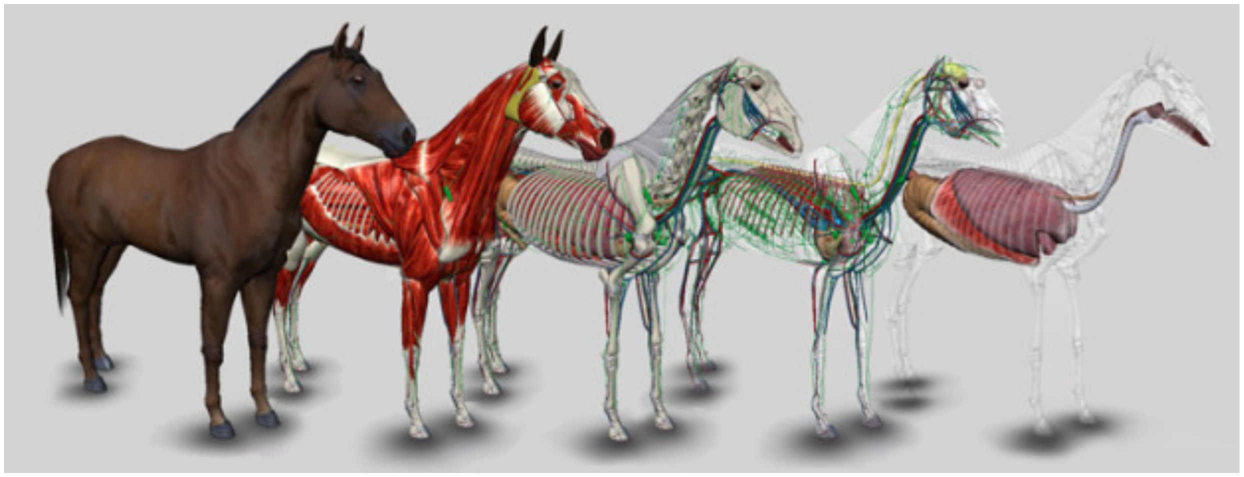
Ngnghm's adventures

Ngnghm's misadventures

Ngnghm's misadventures

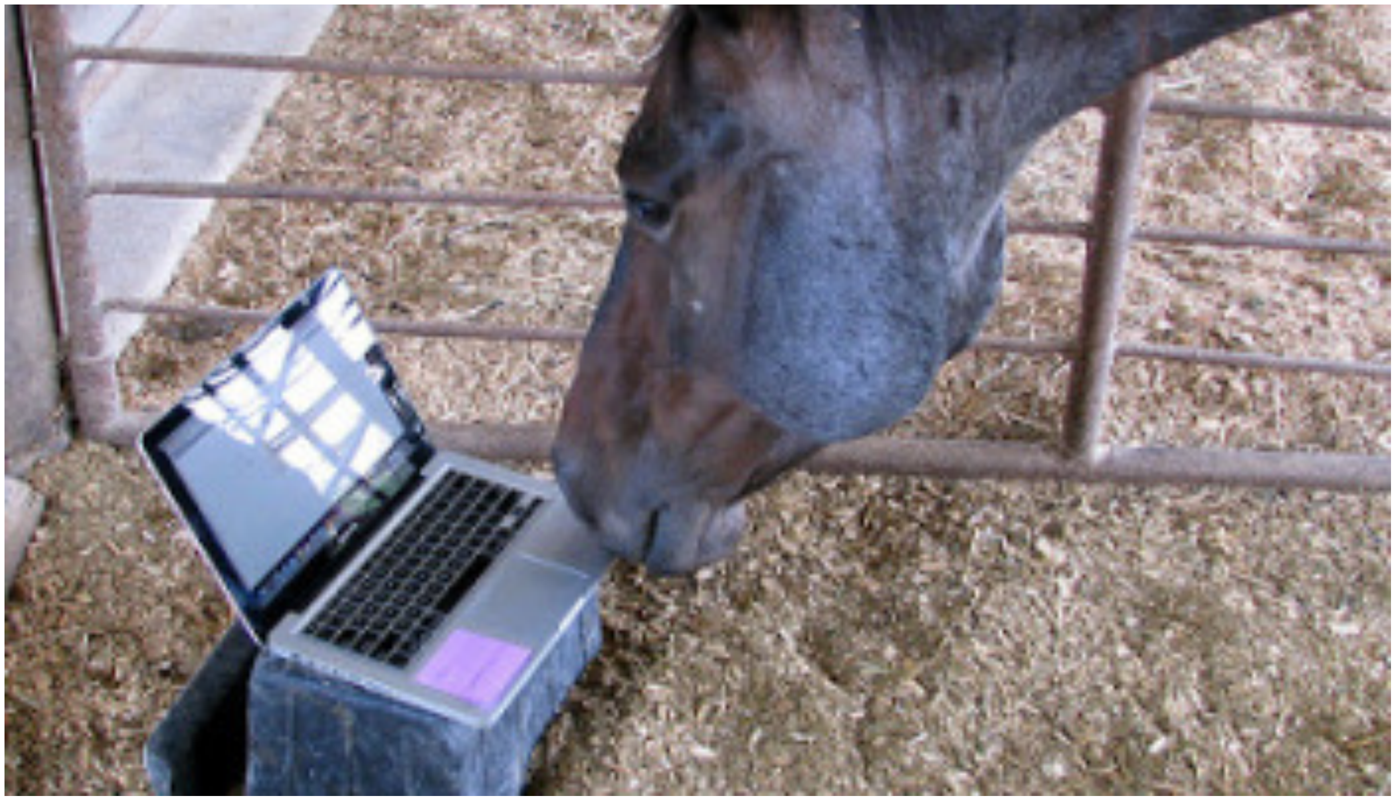


Ngnghm misadventures



Ngnghm saved

Ngnghm hiding



Houyhnhnm Computing

No supernatural magic powers

Just a different point of view

Computing vs Computers

A distinction not about how to write software
but what software to write or not to write

Computing vs Computers

Computers as means, not ends

Computer Science is no more about computers
than astronomy is about telescopes.
— E. W. Dijkstra

Dijkstra: computing as exploring pure maths

Houyhnhnms: maths yes, but foremost, meat

Please Draw me a Sheep (*)

Please Draw me a Sheep (*)

N: Show me a **simple** Human Computing System

Simple Program (*)

```
printf("hello, world\n");
```

C programming language, simple & well formalized

N: It’s full of holes & not even Universal!

Filesystem, editor, compiler, **`make`**, shell, kernel, console, GUI, networking, etc.

N: That’s extremely complex!

Real Simple (*)

Smalltalk world

Alan Kay’s VPRI projects: entire system in 20 KLoC

FORTH, but...

Lower-level makes simpler computer systems,

not simpler computing system

Why Simple? (*)

Reasonable software

Learning

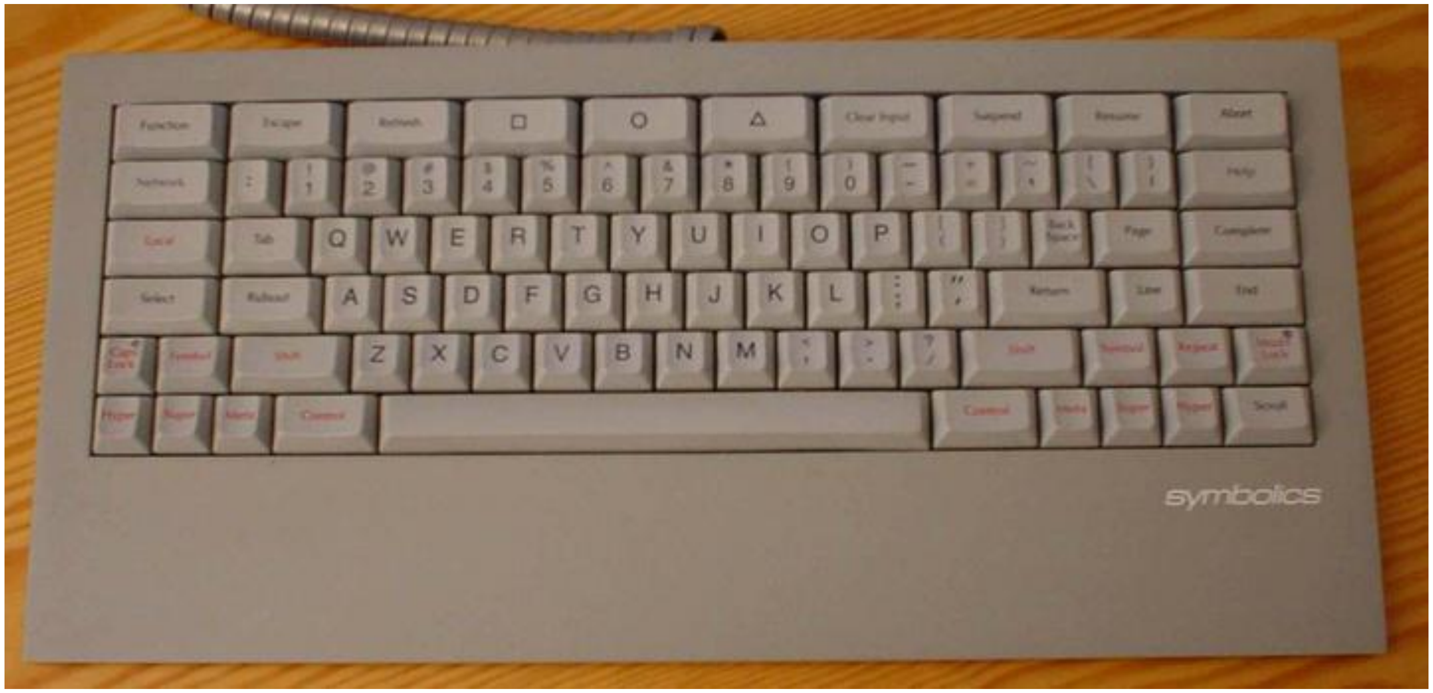
Bootstrapping

Securing

Debugging

Refactoring

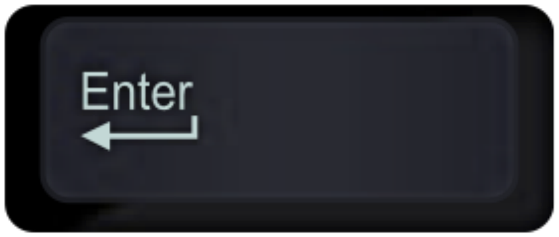
Keyboards



What is this Magic Key Chord?

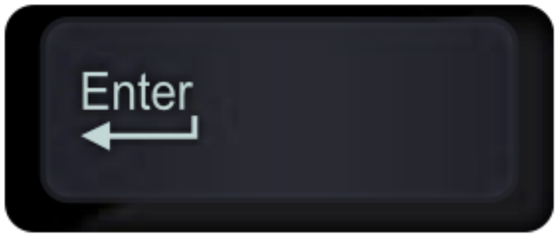
What is this Magic Key Chord?

Not

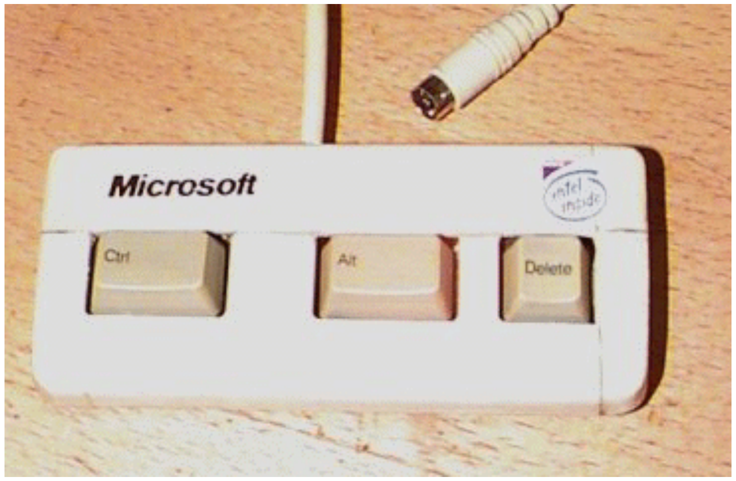


What is this Magic Key Chord?

Not

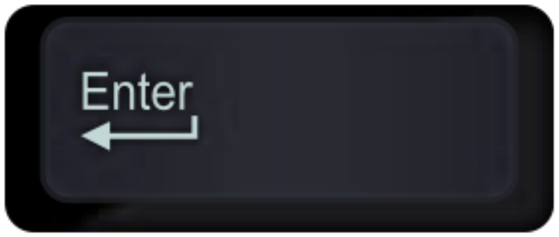


Not



What is this Magic Key Chord?

Not



Not

The magic key chord:

`C-x C-s`

Why Save Files?

Files

If you don’t *save*, all your changes are lost.

Also daily prayer to Baah-Kup, god of data recovery

Protection in the rare and unpredictable event of:

Bad click, closed app, software crash, power loss,

device malfunctioning, lost, stolen, broken, too old...

Why Save Files?

Files

If you don’t *save*, all your changes are lost.

Also daily prayer to Baah-Kup, god of data recovery

Protection in the rare and unpredictable event of:

Bad click, closed app, software crash, power loss,

device malfunctioning, lost, stolen, broken, too old...

Predictable catastrophes, automatable solutions!

Why Save Files Manually?

Historical reasons: big files, small disks

Architecture: that’s how it’s done

Epistemology: can’t autodecide when to save

Personal: been bitten too often

Efficiency: save neurons by saving at every pause.

Why Save Files Manually?

Historical reasons: big files, small disks

Architecture: that’s how it’s done

Epistemology: can’t autodecide when to save

Personal: been bitten too often

Efficiency: save neurons by saving at every pause.

N: Oh, it **was** automated... in **Wetware**!

Sacred Motto

Sacred Motto of the Guild

of Houyhnhnm Programmers:

I object to doing things that computers can do.

— Olin Shivers

Why Compute?

Automate in Wetware?

That’s missing the whole point of Computing!

The highest goal of computer science is

to automate that which can be automated.

— Derek L. VerLee

Wisdom: know to accept defeat

Foolishness: surrender when victory is at hand

II. Orthogonal Persistence

Orthogonal Persistence

Everything is always saved:

If it's not worth saving, it's not worth computing

Do you manually spill registers to caches, to RAM...

Why manually spill RAM to disk, to remote backup?

Persistence Modalities

Latency: persistence after a few seconds max

With world-wide replicas if connected

Space: enough for everything you type or view

Strict determinism → replay from small state

Keep high-level events, discard the irrelevant

Privacy: everything encrypted by default.

Optionally, share with others, or triple encrypt.

Orthogonal Persistence Policy

No one size fits all: latency, replication, privacy...

Programmers can’t possibly **know**

Let **end-users** decide policy tradeoffs.

Define **domains** of persistence.

Make it part of the UI.

Use competing storage providers, world-wide.

No data loss until the End of Civilization.

Houyhnhn Design Principle

When designing interaction for data entry, configuration, etc:

Each can specify what they know,

none need specify what they don't

Fractal Transience

Modern Apps may superficially save for you

But it’s not reliable, replicated, etc.

Underneath, it’s transience at every level

Complex protocols for serializing, saving, transacting, upgrading, converting, etc.

End-programmers can’t handle all configurations

Responsibility pushed back to helpless end-users

Disaster of Homeric Proportion

Joke on me for compulsively typing **`C-x C-s`** ?

Disaster of Homeric Proportion

Joke on me for compulsively typing **`C-x C-s`** ?

Joke on N!

Since rescue, used Microsoft Paint to take notes

Her travels & travails, marvels & ordeals... in verse

How many Homeric Sagas have been lost?

How many thousand man years wasted world-wide?

Some Perks of Orthogonal Persistence

Never lose your session to a crash

Not just tabs, but *all* session state

Infinite Undo/Redo... for free!

Eternal instant replay

Others can’t delete data on you

You own (search, analyze) your past data

One disadvantage

Deletion is costly

Cost linear in amount of data since item to delete

Irreversible: requires a lot of testing

Limitation: Other people keep their copies

De-indexing is almost free!

One disadvantage

Deletion is costly

Cost linear in amount of data since item to delete

Irreversible: requires a lot of testing

Limitation: Other people keep their copies

De-indexing is almost free!

Primitives: Immutable facts, not mutable records

Perks for Programmers

Write at least 30% less code

No "save" primitives — Yes "Don’t save" primitives

Experiment without fear

Bugs 100% reproducible by default

Think in terms of your abstractions, not of bits

Don’t implement from "the bottom"

Yahoo Metaprogramming

Programs: specify everything "from the bottom"

"bottom": abstractions common to all end-users

Metaprograms: most are compile-time only,

some are horrible runtime kluges

Leaky Abstraction: programs see what’s underneath

Brittle conventions. Security issues.

Houyhnhnm Metaprogramming

Programs: specified from as high-level as possible

But not higher. Write each program in its own DSL.

Metaprograms: what is *underneath*

Implement the DSL.

Full Abstraction: programs can't see what's under

(except through explicitly allowed non-determinism)

Program Migration

Change underlying metaprogram without restarting

Change policy: I/O devices (console, sound...), persistence, speed, latency, robustness, "optimization level", privacy, trust, logging, omniscient debugging, etc.

Garbage Collection **is** migrating the object graph

Endless possibilities...

Everything is different!

No impermanence (but cleanup discipline)

No byte sequences (but object boundaries)

No saving files (but release management)

No filenames (but data indexing)

No startup scripts (but config diffs)

No finite access rights (but domain capabilities)

Not source but semantics (*)

Yahoos: byte sequences, with explicit encoding;

text files as source, or worse;

either unstructured or structured editing.

Houyhnhnms: encoding-independent objects;

semantics as source, auto encoding migration;

editable views, both unstructured or structured

Bad Memories

What if you hose your system? Grave errors persist!

OOM, fork bomb, deep metabug, garbled store, etc.

Try not to!

Keep old snapshots, drop irrelevant data, use watchdogs, regular drills for backup restoration...

Yet what if you do? (Old Eumel story...)

Bad Memories

What if you hose your system? Grave errors persist!

OOM, fork bomb, deep metabug, garbled store, etc.

Try not to!

Keep old snapshots, drop irrelevant data, use watchdogs, regular drills for backup restoration...

Yet what if you do? (Old Eumel story...)

Escape to the meta-system!

III. Reflective Architecture

The Meta System

The Monitor: simple system with reserved memory

Can merge in parts of the system being debugged

Reason to build complex systems from simple ones

What about bugs while debugging?

Virtualize it! (Thus reserve *virtual* memory, too)

Dichotomize where the catastrophe happened

Real Scary Bugs

A metabug breaking persistence can wreak havoc

Can’t completely avoid *all* bugs, yet can eliminate entire classes: prove correctness, test like SQLite

Most Houyhnhnm programs don’t touch persistence

At least: not worse than for Yahoos

Fallback: religiously keep (and test) old backups

Virtualization as Branching

Entire system under **`git`** —— but live, not dead

Virtualization **is** branching

Undoing bad bug is rebasing (orwellian retcon)

Limits: can’t unprint the data, unlaunch missiles

Hardware acceleration welcome, not needed

What **is** needed: **Full Abstraction**

Full Abstraction (Again)

Higher levels can’t see implementation specifics

Essential difference b/w Houyhnhnms and Yahoos

Property of *system*: not hardware, not language

Leaky abstractions are not just top priority bugs:

Full Abstraction (Again)

Higher levels can’t see implementation specifics

Essential difference b/w Houyhnhnms and Yahoos

Property of *system*: not hardware, not language

Leaky abstractions are not just top priority bugs:

They are security issues.

Call cops on abstraction breakers

First-class Computing Systems

First-class Computing Systems

Not just programming language

Includes runtime execution state

Every program is an abstract computing system

Not a One True Global Computing System

Abstract over incremental computing system diffs

Virtualization not just for sysadmins

http://j.mp/BHfci2016

Lego Stacks, Not Rigid Towers

Yahoo apps: rigid towers rooted in the OS

Houyhnhm components: stackable modules

Yahoo Operating Systems: privileged kernels

Houyhnhnm: dynamically relinking metaprograms

Rigid is Fragile (*)

Each leaky abstraction is a costly liability

Error situations bubble up to the user

Manual parsing, unparsing: slow, brittle, unsecure

Validate inputs → *Language-Theoretic Security*

Compatible? Find common tower level

Adapters **above** lose atomicity

→ Need common metalevel **below**

Quality in Human Software

Persistence, Robustness, Security, Upgradability, Maintainability, Understandability, Explorability...

Dismissed by Yahoo programmers as "Non-functional requirements" of their artifacts

To Houyhnhnms, they essential aspect of computing interactions

Choosing artifacts over interactions:

Sacrificing ends to means

Meta-level components (*)

Compression algorithms (lossless vs lossy)

New synchronization algorithm (e.g. CRDT)

Optimization passes (safe or unsafe)

Data representations (depending on constraints)

Solve issue once with a meta-program

Not every time manually with a "design pattern".

Turtling down

Infinite turtles below...

and you can change them, dynamically!

Building software *down*, not just *up*.

That’s why neither libraries nor servers will do.

No Kernel

Initial resource handler → Bootstrap chain

Access control → Transferable ownership

Resource management → Linear Logic

Multiplexing access → Servers

Implementing services → Meta-programs

Virtualization → First-class full Abstraction

No Apps

One-way communication? Documents, not apps

Both ways? Extensible Platforms (like Emacs)

Yahoos: app each reimplement own bad selector

Houyhnhnms: component solves issue for everyone

Each focuses on his specialty (division of labour)

Human Precedents (*)

T diagram: translate from A to B, running in C

Lisp: great at abstraction, but leaky as hell

PLT Racket: PL construction kit

Getting There (*)

Who to pay costs of Migration?

How to keep up with Yahoos?

How to win despite network effects?

How not to fall into autism?

Where to bootstrap from?

No worries

Metaprogramming: programming programming

Vision and Will come foremost

If you want to build a ship, don't
drum up the men to gather wood,
divide the work, and give orders.
Instead, teach them to yearn
for the vast and endless sea.
— Antoine de Saint-Exupéry (creatively misquoted)

Know the Goal

The goal is not a system

The goal is a saner approach to computing

You can already do it, today, everyday

Living as a Houyhnhnm, Not a Yahoo (*)

Apply the knowledge. Share it with colleagues.

Change the project.

Not every project can be reformed.

Sometimes, restart from scratch.

Sometimes, quit the project, or even the company.

You can't save other people.

But you can save yourself.

There is no path to Happiness.

Happiness is the path.

Yet recognize the tree by its fruits.

Where to Start? (*)

Start Anywhere: not just from the "bare metal"

Start Simple: no "all or nothing". Systemantics.

Do your thing: division of labor

Build bridges: division of labor

Join forces: division of labor

Go down, not just up

Ideas have consequences (*)

New PoV leads to new architecture

Interaction isn’t one-way command-and-control

The Take Home Points (redux)

Interactions around us, not artifact below us

Orthogonal Persistence

Reflective Systems: First-class Computation

A change in point of view changes architecture

Challenge: Condense the Vaporware

Bring Runtime Reflection to your platform

Platform: PL, IDE, OS, Shell, Distributed System

First-class implementations, meta-levels, migration

Enjoy simplification, persistence, robustness, etc.

The Meta-Story

More ambitious than just software:

The Meta-Story

More ambitious than just software:

A change of point of view about computing

Not mere vaporware: *wetware*

The Meta-Story

More ambitious than just software:

A change of point of view about computing

Not mere vaporware: *wetware*

Thanks!

Houyhnhnm Computing:

http://ngnghm.github.io/