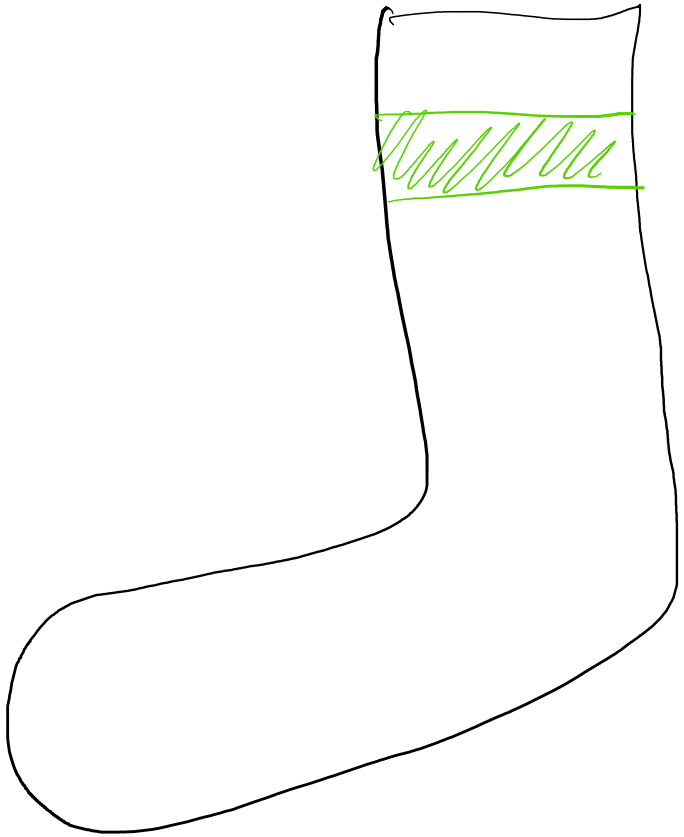
Warmup exercise



On the left we see:

1. A green sock
2. A white sock
3. A white sock with a green stripe

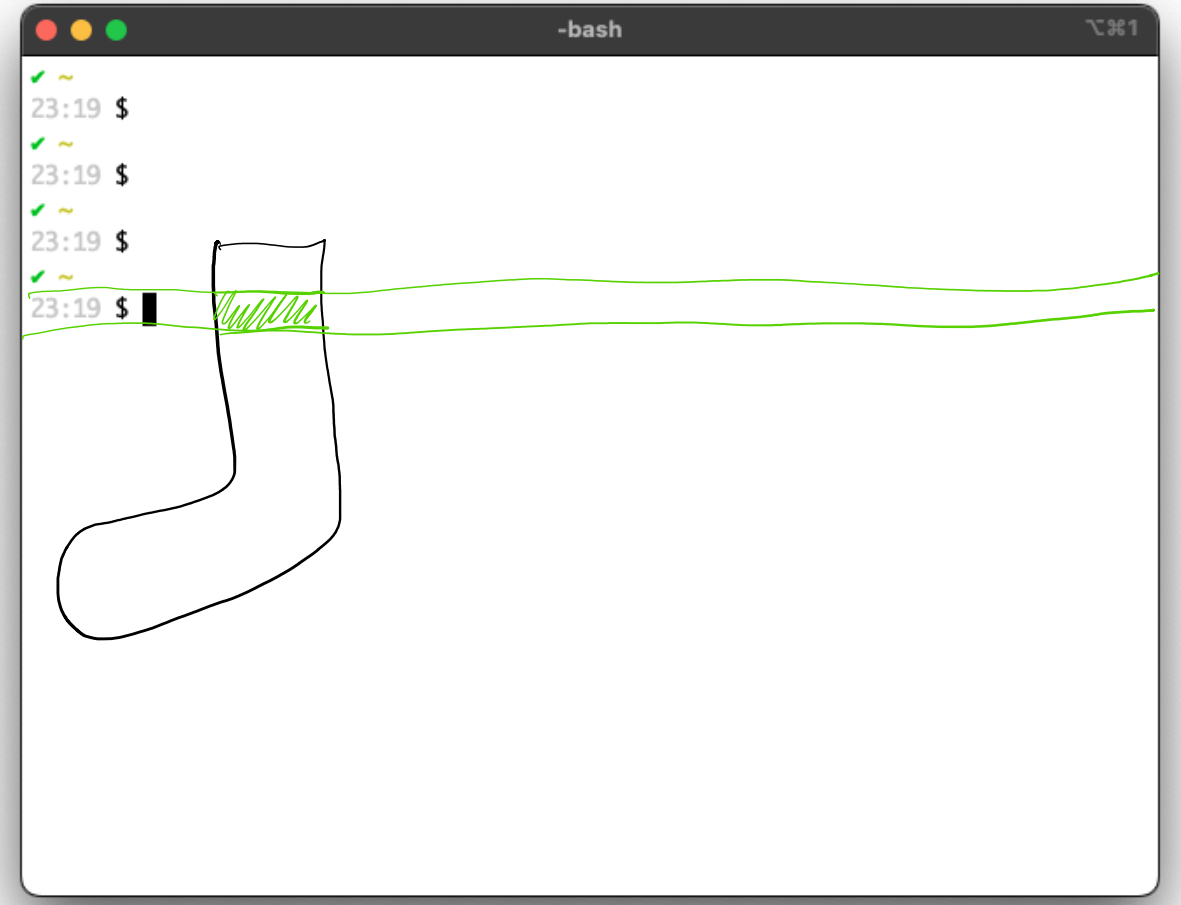
https://github.com/ngs-lang/presentations

Warmup exercise

On the left we see:

1. Interactive shell
2. Non-interactive shell
3. Slightly interactive shell

Warmup exercise



On the left we see:

1. Interactive shell
2. Non-interactive shell
3. Slightly interactive shell

Hint: is that a green sock?

Hint: interactive shell vs interactive website

Unix Shell

We can do better now

Command Line Interface is not the pinnacle of UI today

About Me

- Ilya Sher
- Bash user – since around 1996
- DevOps & Software – professionally since 2001
- Working on Next Generation Shell – since 2013

CLI – how we got here?

Very old communication paradigm:

send text – receive text

CLI – how we got here?

- Telegraph
- Teleprinter (aka teletype, aka TTY)
- Video Display Unit

CLI – how we got here?

- Telegraph
- Teleprinter (aka teletype, aka TTY)
- Video Display Unit

CLI – how we got here?

- Telegraph
- Teleprinter (aka teletype, aka TTY) - with computer
- Video Display Unit

CLI – how we got here?

- Telegraph
- Teleprinter (aka teletype, aka TTY)
- Video Display Unit

Turning Point

Technological breakthrough – VT52 with cursor movement support

Allowed full screen user interfaces

1. 1971 - 1975
2. 1976 - 1980
3. 1981 - 1985

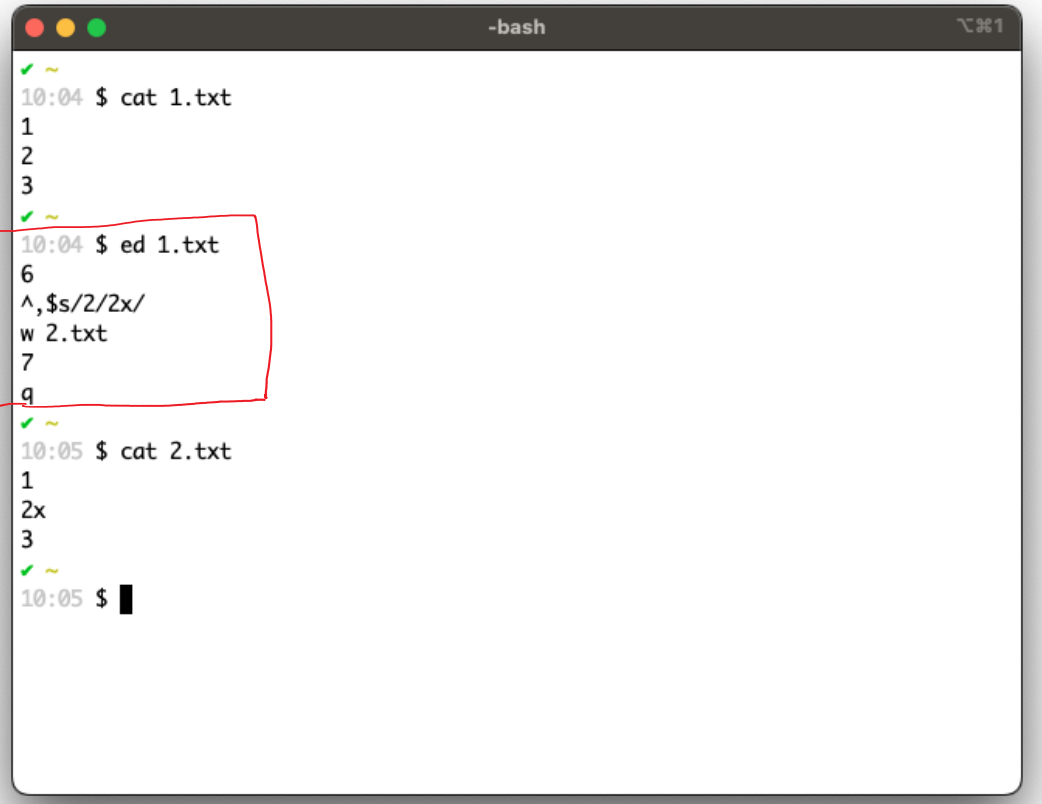
Turning Point

Technological breakthrough – VT52 with cursor movement support

Allowed full screen user interfaces

1. 1971 - 1975 (answer: 74/75)
2. 1976 - 1980
3. 1981 - 1985

Turning Point – Text Editing



Before: CLI text editing

Text editing goes full screen instead of CLI.

Bill Joy releases vi in 1976

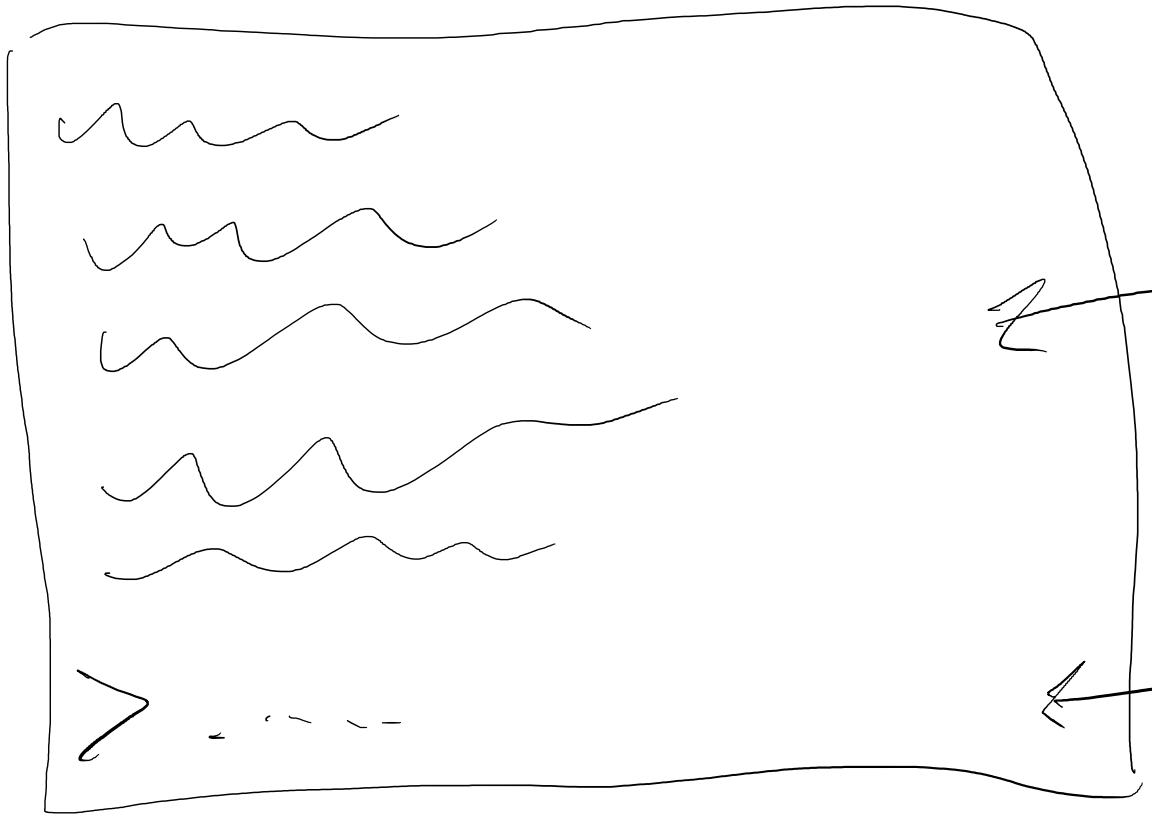
After: full screen text editing

Turning Point - Shells

Not a turning point

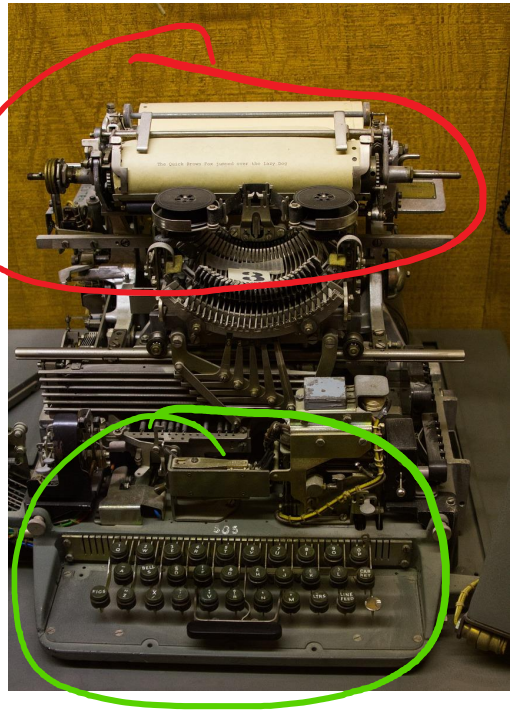
Shells never responded (except for command line completion)

User Interface - “Interactive Shell”

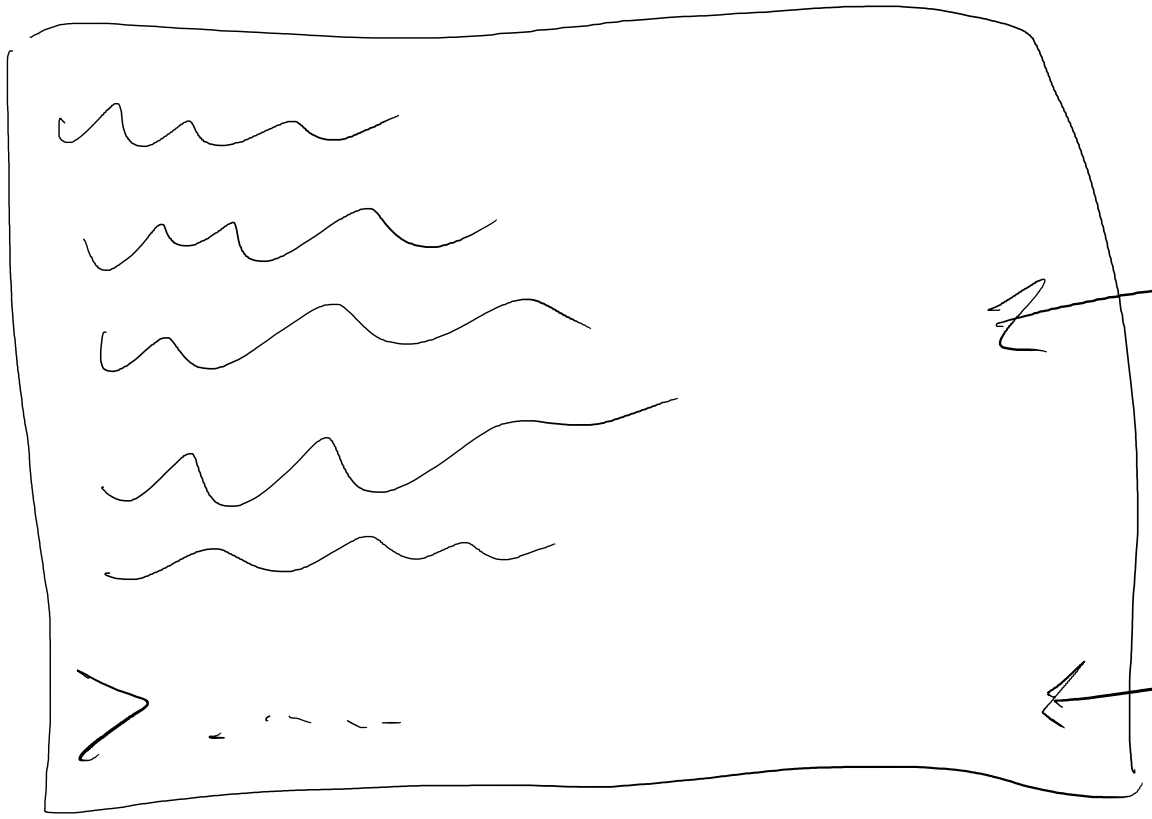


Most of the screen
Not interactive,
paper-like

One line, the
“command line”
Interactive



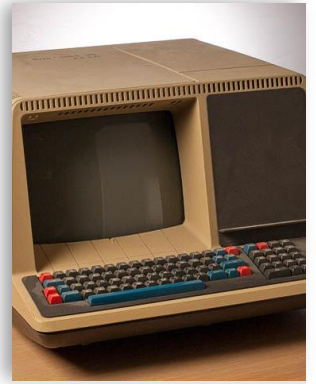
User Interface - “Interactive Shell”



Most of the screen
Not interactive,
paper-like

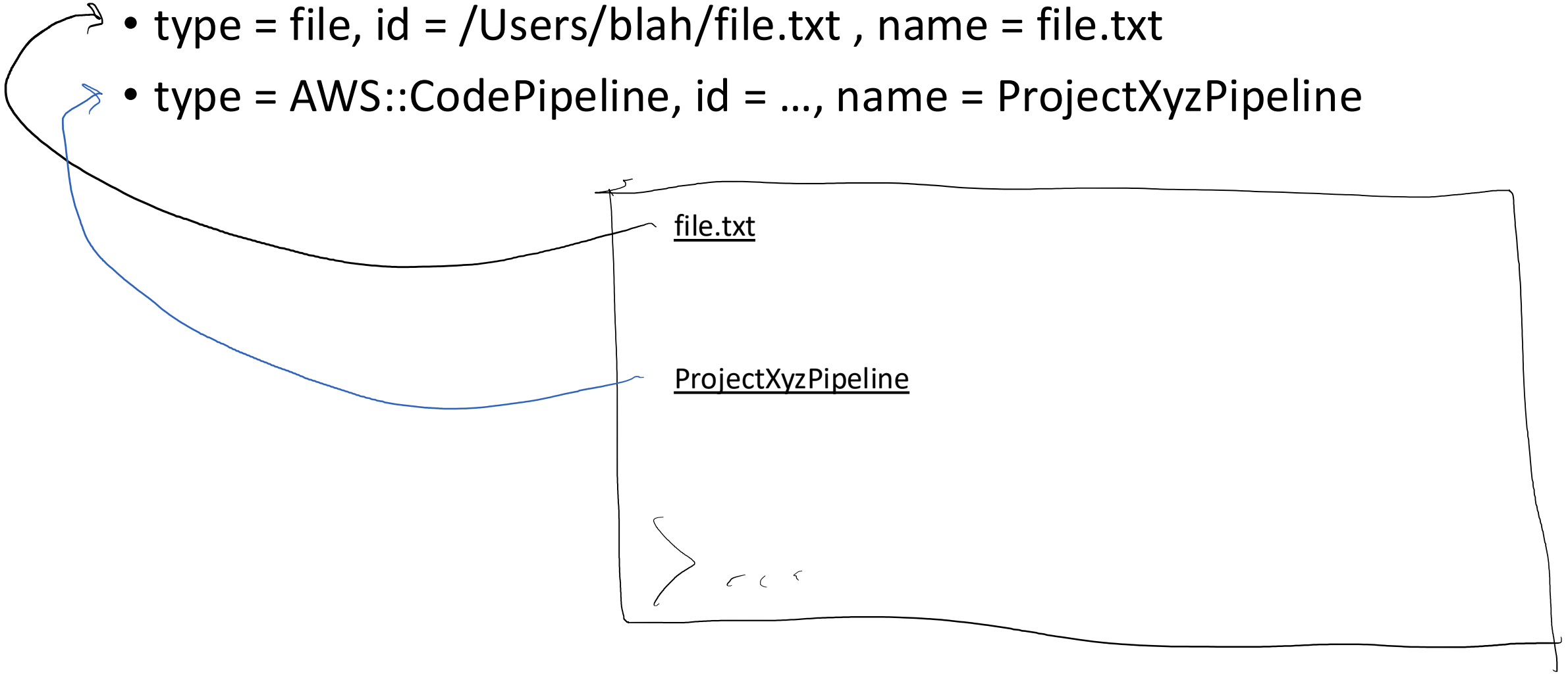
Should be interactive

One line, the
“command line”
Interactive



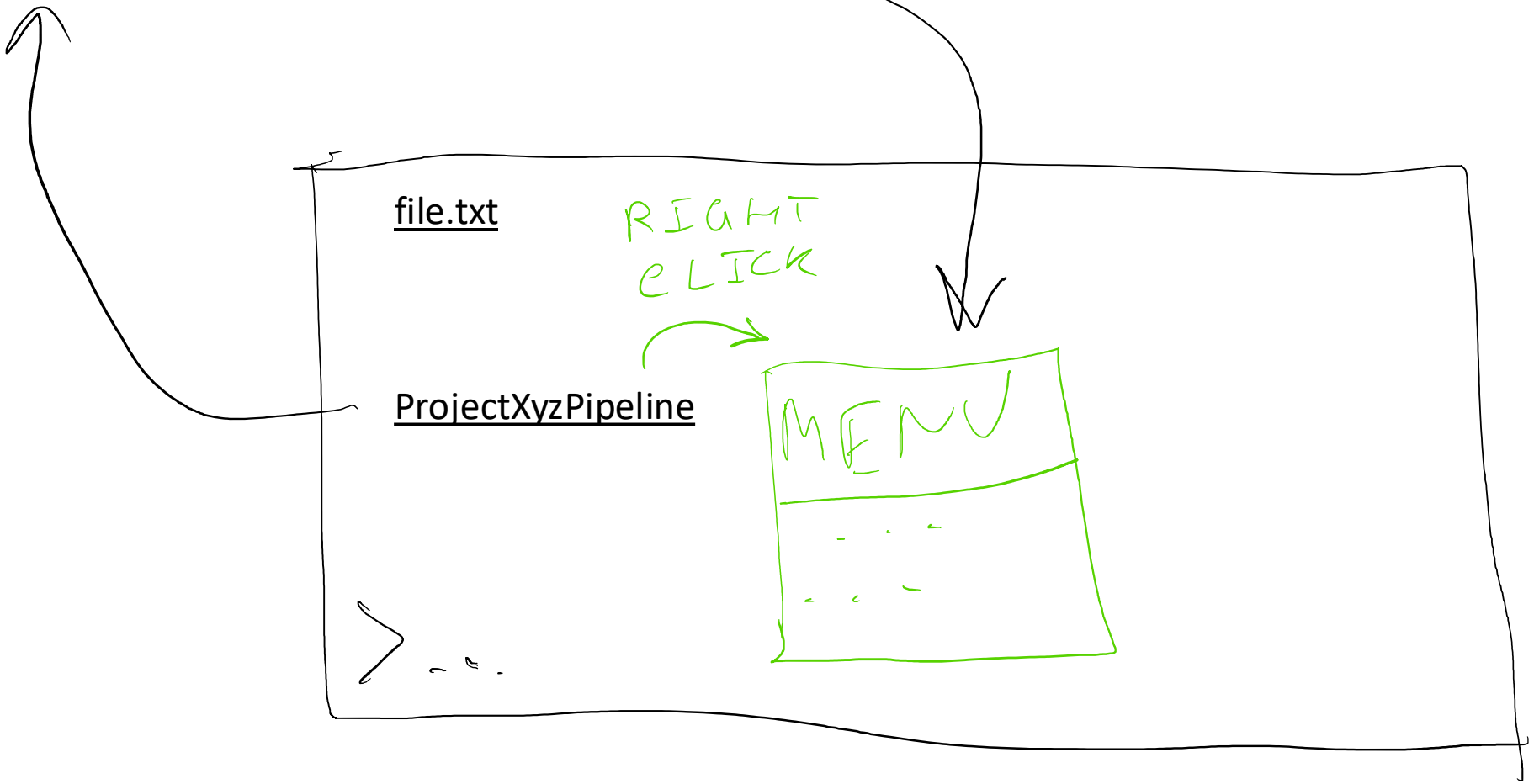
User Interface – Objects on the Screen

- type = file, id = /Users/blah/file.txt , name = file.txt
- type = AWS::CodePipeline, id = …, name = ProjectXyzPipeline

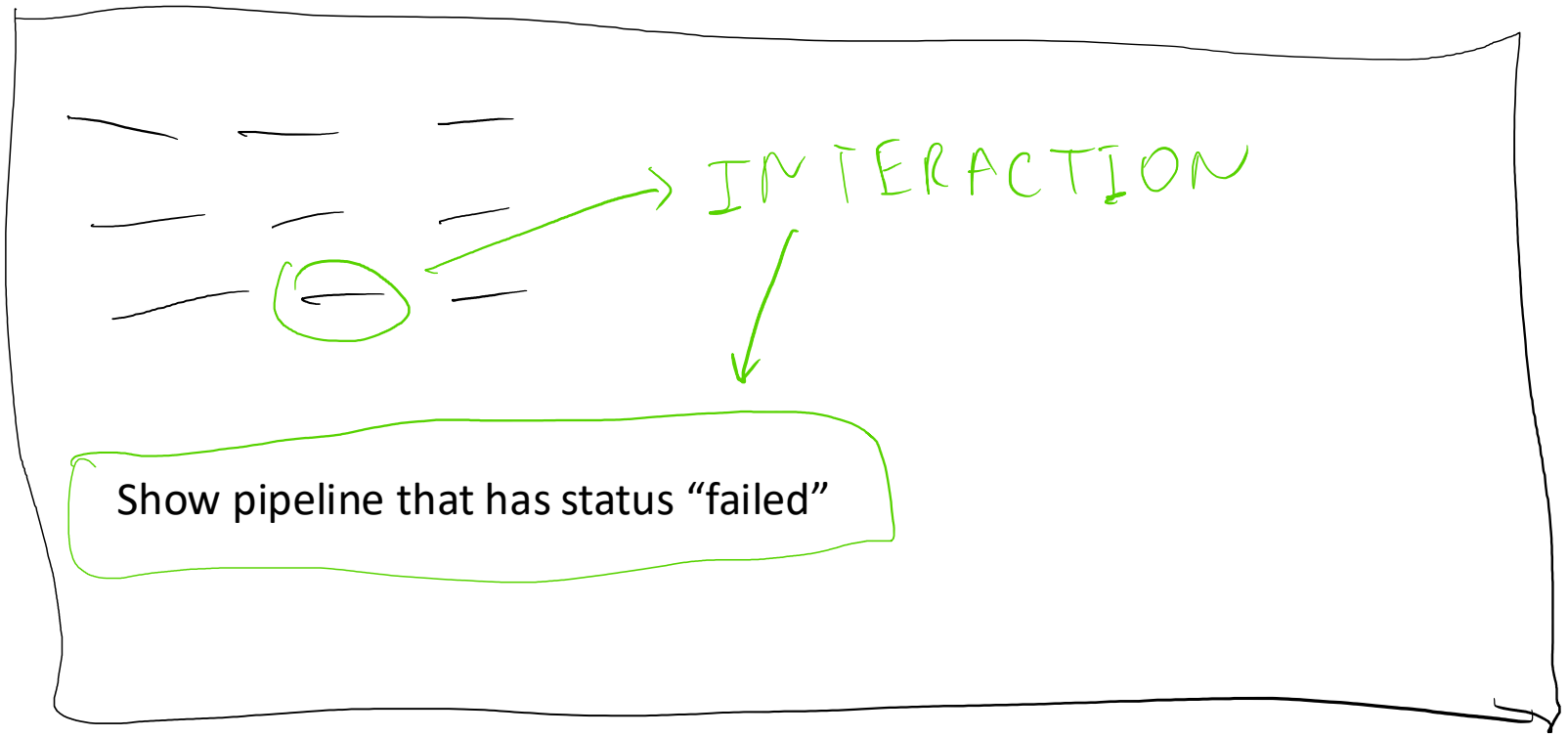


User Interface – Plugins

- Plugin Blah: handles AWS::CodePipeline



User Interface – Record / Replay

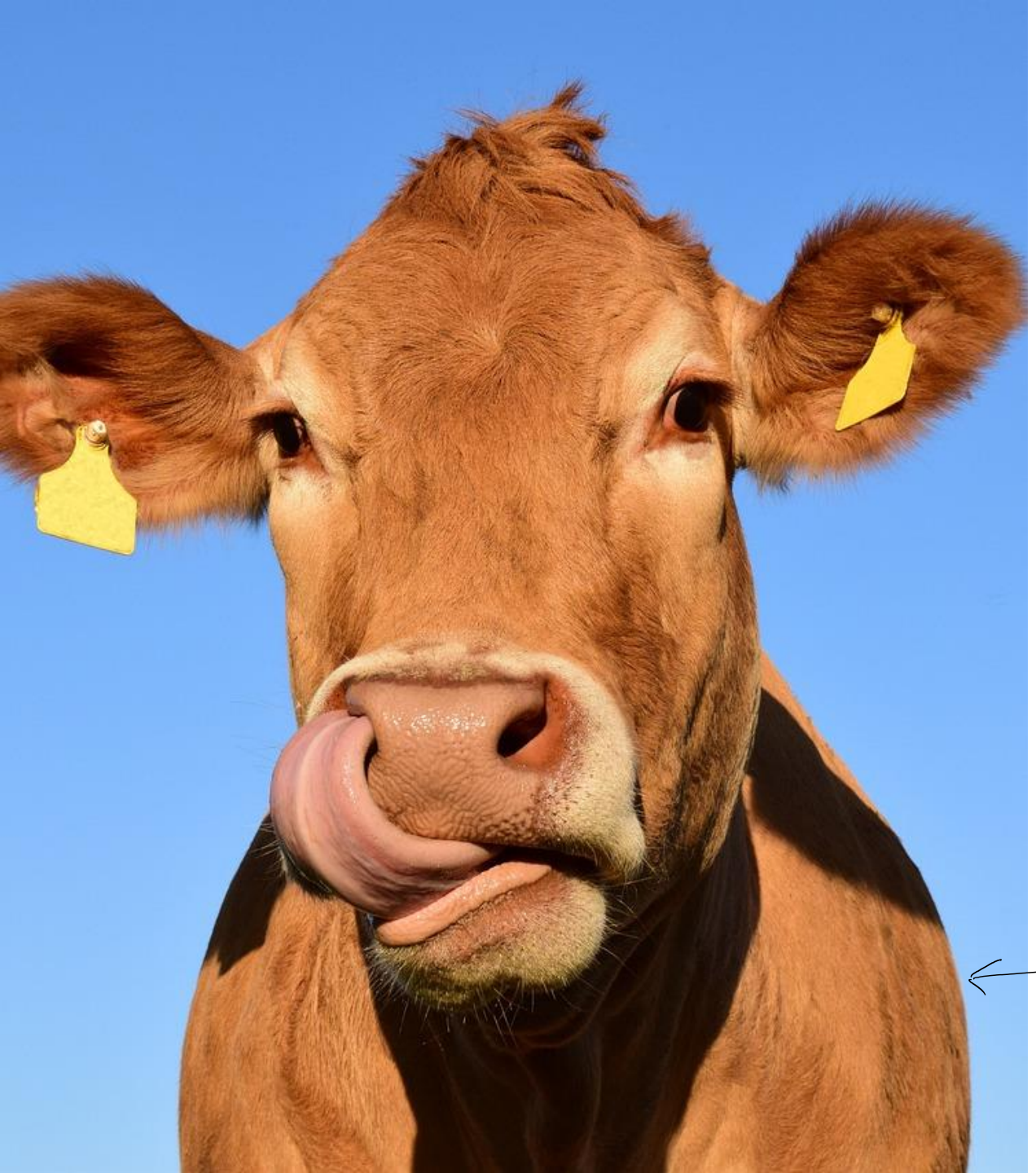


Capture the semantics of the interaction

More powerful than just history: i-123

Display the recording

Let edit the recording

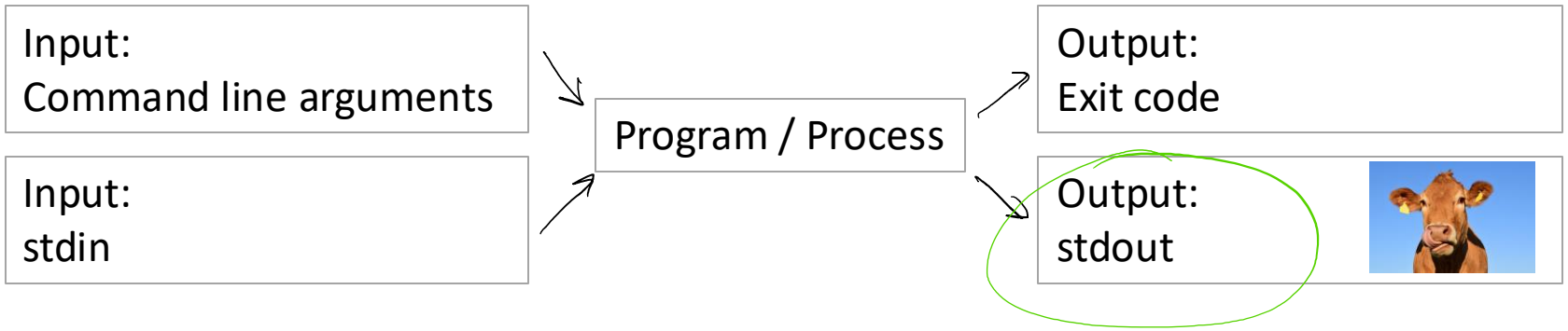


User Interface – How?

SACRED COW

User Interface – How?

Understanding Output



"Shell is not supposed to get into semantics"

User Interface – How?

Understanding Output

Semantics – already implemented

Command line completion

Input:
Command line arguments

Input:
stdin

Program / Process

Semantics – already implemented

Basic error handling

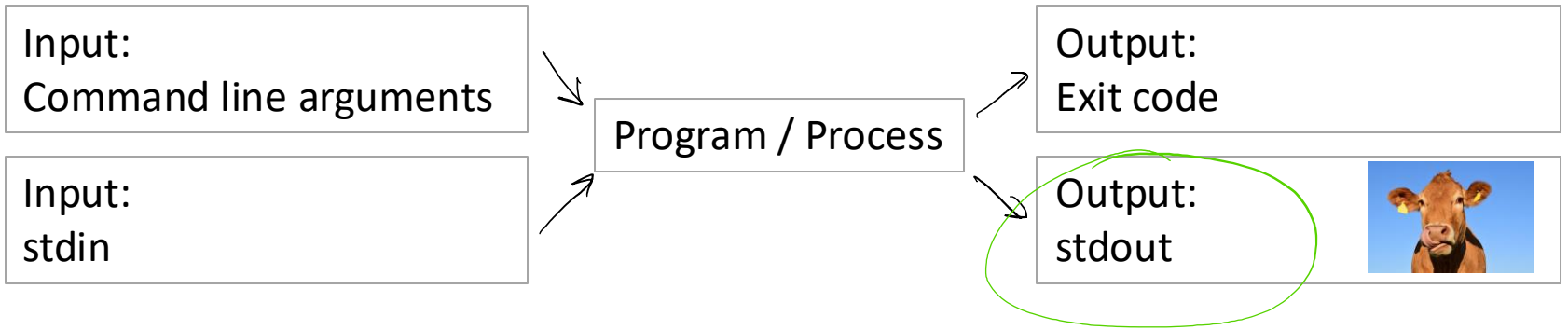
Output:
Exit code

Output:
stdout

"Shell is not supposed to get into semantics"

User Interface – How?

Understanding Output



"It's too much work"

User Interface – How?

SAME SCALE

Implemented in shell for many commands

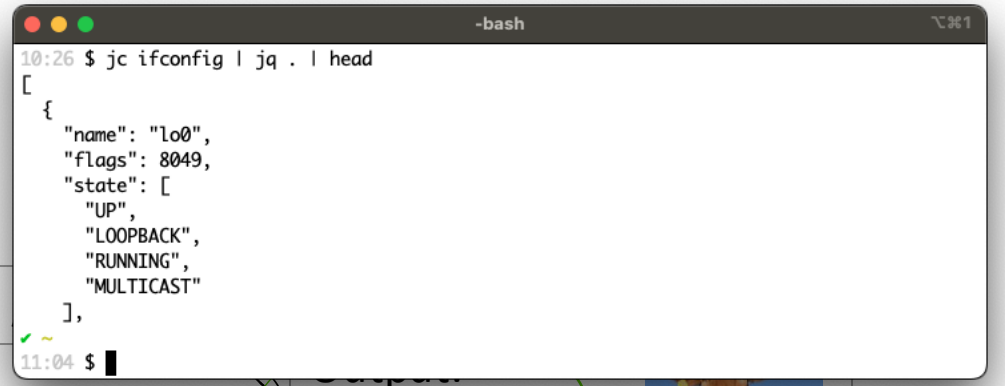
Command line completion

Input:
Command line arguments

Input:
stdin

Program

Understanding Output



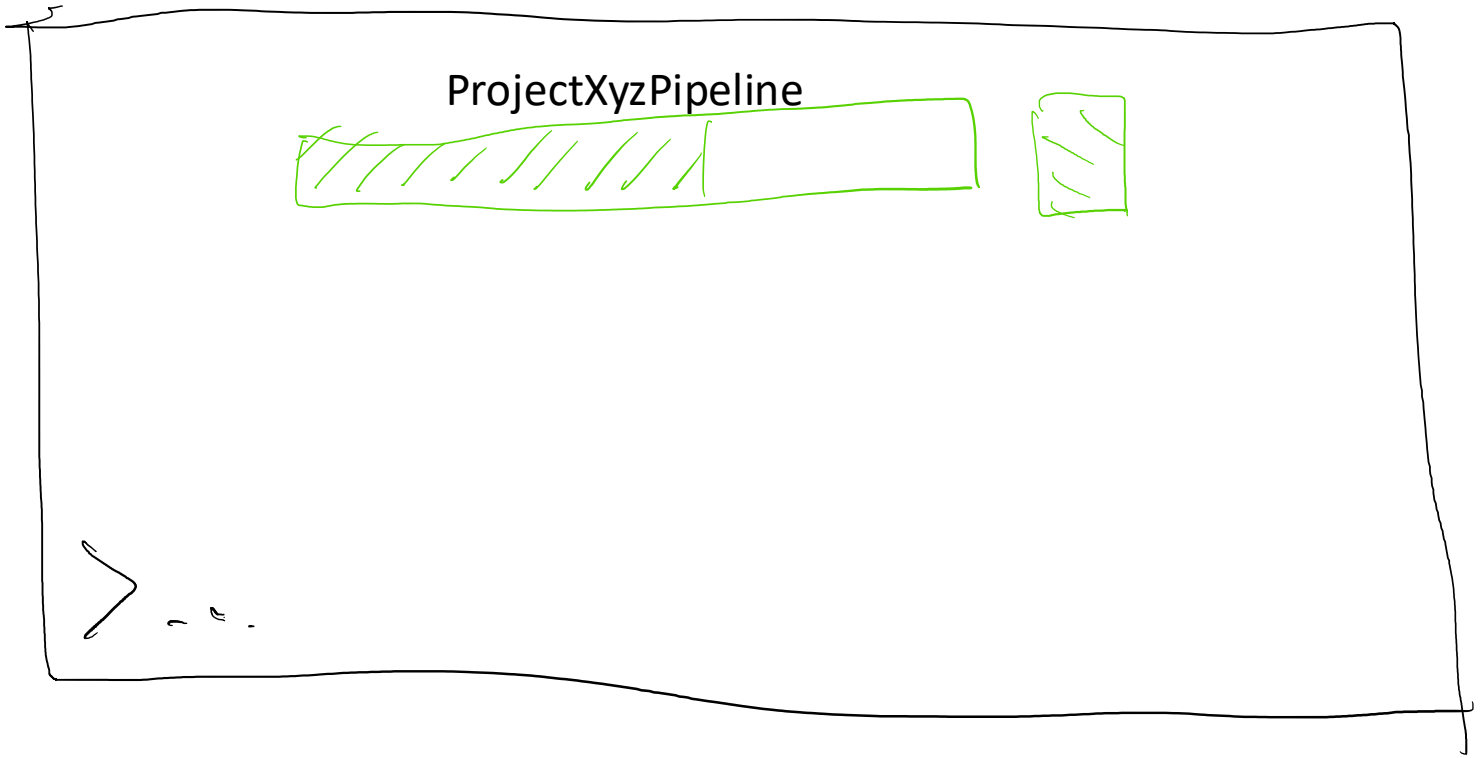
stdout

"It's too much work"

User Interface – Objects of Interest

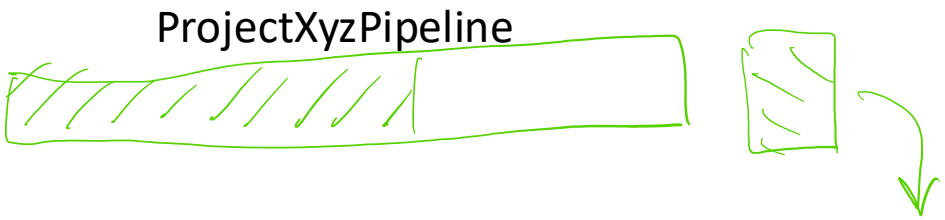
- Information displayed automatically – ex: CI/CD progress

Would be very awkward if implemented in other shells



User Interface – Objects of Interest

- Information displayed automatically – ex: CI/CD progress



Sensible default rules like:

- Show CI/CD runs of pipelines that I created
- Show CI/CD runs that were triggered by my commit

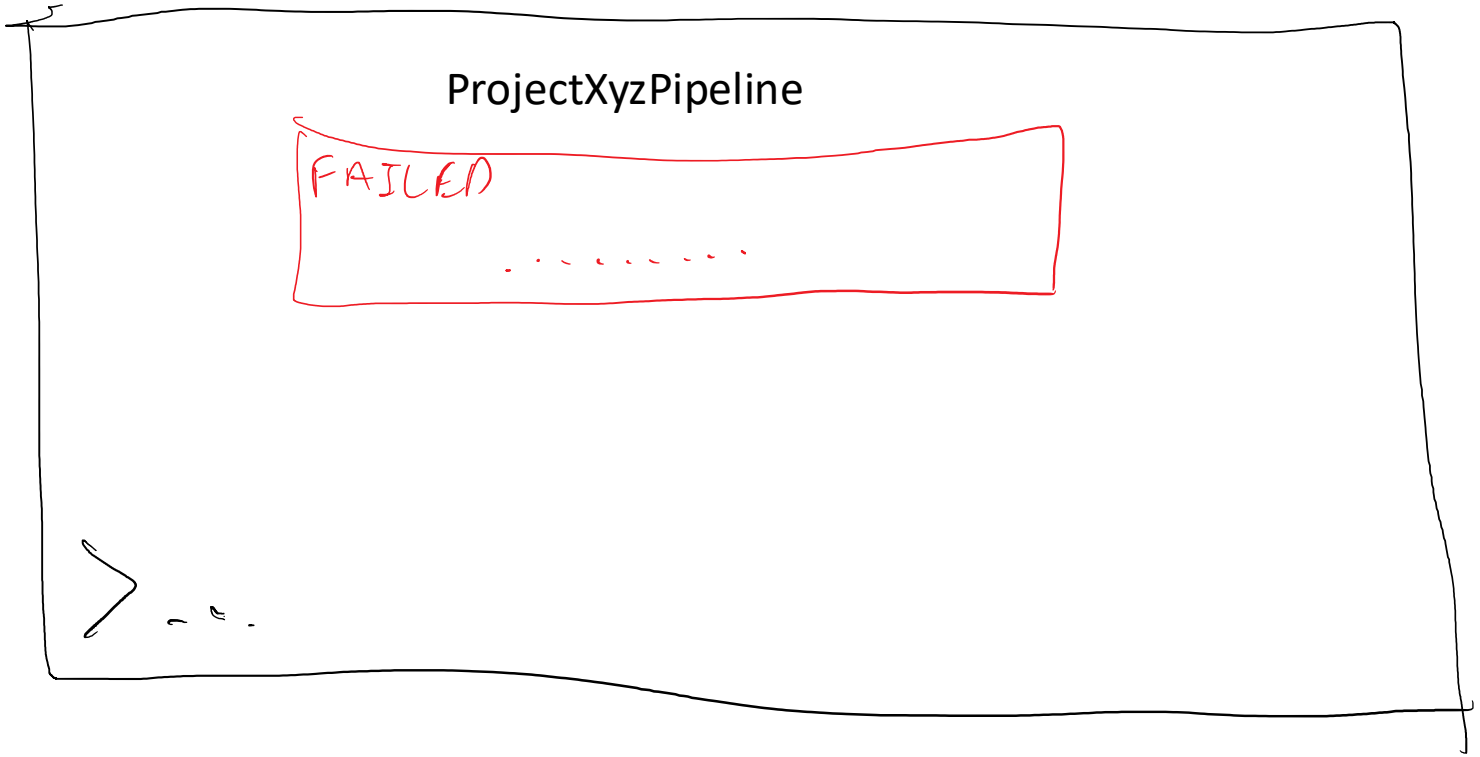
Same but for last errors instead of runs

This information is shown to you because it matches rule "blah"

- *Edit rule*
- *Delete rule*
- *Exclude this object*

User Interface – Objects of Interest

- Information displayed automatically – last error



UI Vision Summary

Semantics - the more a program "understands" the more powerful it can be.

- Interactive objects on the screen

- Capture as much of interaction semantics as you can.

Thanks!

- Next Generation Shell - https://ngs-lang.org/
- Ilya Sher - https://ilya-sher.org/

What's next?

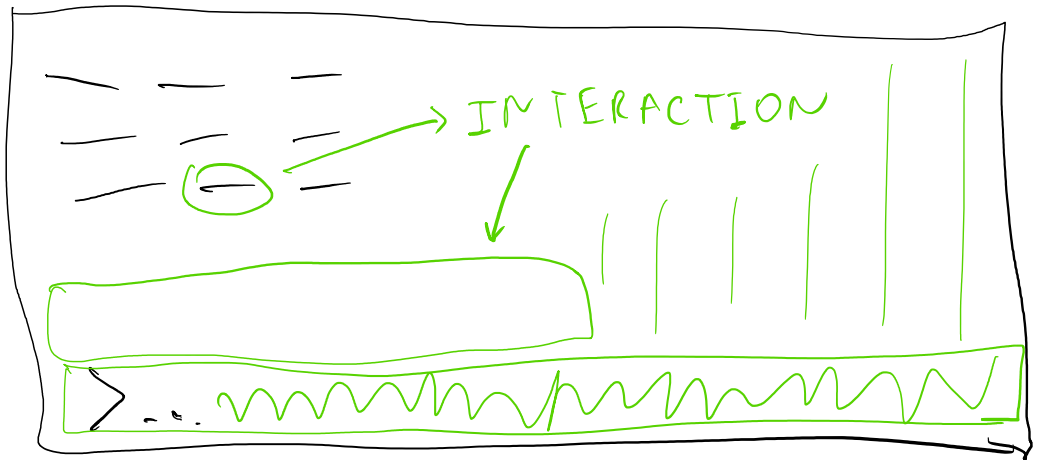
- Talk to me / join Discord / give feedback
- Try NGS
- Spread the word
- Help designing the UI

https://github.com/ngs-lang/presentations

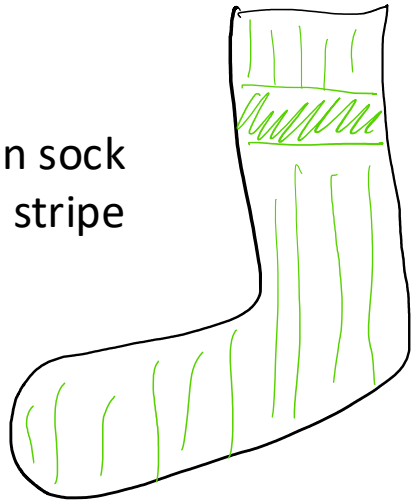
Bonus Slides

Interfaces

	Web	CLI
Pros	Easy	Powerful
Cons	• No history • Non-repeatable • Limited	• Requires programming to achieve almost anything (or it won't be reproducible) • Can't interact with the output



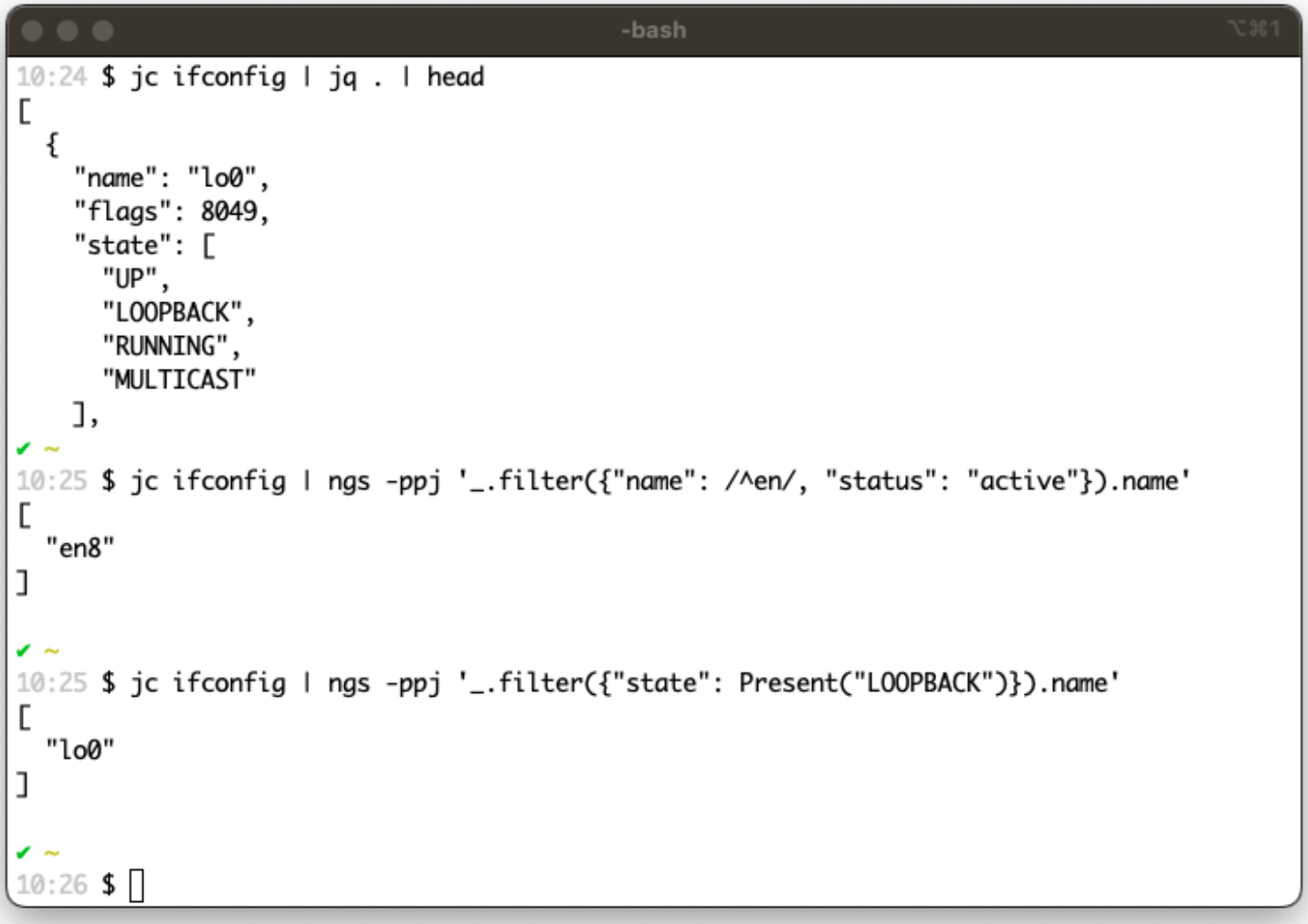
Green sock
… with green stripe



NGS Architecture

Component	Implemented in
UI frontent – thin plugins	JS / NGS
UI backend	NGS
NGS stdlib	NGS
NGS language core	C

jc with NGS


-bash
10:24 $ jc ifconfig | jq . | head
[
{
"name": "lo0",
"flags": 8049,
"state": [
"UP",
"LOOPBACK",
"RUNNING",
"MULTICAST"
],
10:25 $ jc ifconfig | ngs -ppj '_.filter({"name": /^en/, "status": "active"}).name'
[
"en8"
]
10:25 $ jc ifconfig | ngs -ppj '_.filter({"state": Present("LOOPBACK")}).name'
[
"lo0"
]
10:26 $

Thanks!

- Next Generation Shell - https://ngs-lang.org/
- Ilya Sher - https://ilya-sher.org/

What's next?

- Talk to me / join Discord / give feedback
- Try NGS
- Spread the word
- Help designing the UI

https://github.com/ngs-lang/presentations

Thanks!

https://github.com/ngs-lang/presentations