NuShell

Superglue for your OS

Willem Vanhulle

SysGhent

Wednesday, Dec 3, 2025

Introduction

What does the following Bash code do?

```shell
find . -type f -name "*.log" -mtime +30 -exec rm {} \;
```

What does the following Bash code do?

```shell
find . -type f -name "*.log" -mtime +30 -exec rm {} \;
```

Nu:

```shell
ls **/*.log | where modified < (date now) - 30day | rm
```

Improvements:

- Decomposes the problem with pipes
- Does not require `find` flags
- Built-in glob, duration and date type

What does Nu in NuShell stand for?

What does the following Bash code do?

```shell
find . -type f -name "*.log" -mtime +30 -exec rm {} \;
```

Nu:

```shell
ls **/*.log | where modified < (date now) - 30day | rm
```

Improvements:

- Decomposes the problem with pipes
- Does not require `find` flags
- Built-in glob, duration and date type

What does Nu in NuShell stand for?

New shell.

Prerequisites

Try it out out yourself

No installation: https://www.nushell.sh/demo/

Install locally: https://www.nushell.sh/book/installation.html

- Download binary: github.com/nushell/nushell/releases
- Rust:
  - Install `rustup` from https://rustup.rs/
  - Add Cargo bin to your PATH if not done automatically
  - `cargo install nu`
- Mac: `brew install nushell`
- Windows (winget): `winget install nushell`
- Windows (chocolatey): `choco install nushell`

Linux:

- Debian: `apt install rustup`
- Nix: `nix-shell -p nushell`
- Snap: `sudo snap install nushell --classic`

Exercises

Have a look at the `*.nu` files in this repo.

To run an exercise:

```shell
workshop.nu # With shebang
nu workshop.nu
```

To pipe in data from Bash:

```shell
cat somefile.txt | exercise.nu # With shebang
cat somefile.txt | nu exercise.nu
```

Piping within NuShell:

```shell
open somefile.txt | exercise.nu
```

Basics

Commands output tables

Shell

```
ls
# => ╭────┬──────────────────┬──────┬──────────┬──────────────╮
# => │  # │       name       │ type │   size   │   modified   │
# => ├────┼──────────────────┼──────┼──────────┼──────────────┤
# => │  0 │ CITATION.cff     │ file │    812 B │ 2 months ago │
# => │  1 │ CODE_OF_CONDUCT.md │ file │  3.4 KiB │ 9 months ago │
# => │  2 │ CONTRIBUTING.md  │ file │ 11.0 KiB │ 5 months ago │
# => │  3 │ Cargo.lock       │ file │ 194.9 KiB │ 15 hours ago │
# => │  4 │ Cargo.toml       │ file │  9.2 KiB │ 15 hours ago │
# => │  5 │ Cross.toml       │ file │    666 B │ 6 months ago │
# => │  6 │ LICENSE          │ file │  1.1 KiB │ 9 months ago │
# => │  7 │ README.md        │ file │ 12.0 KiB │ 15 hours ago │
# => ...
```

Sort output by column


```
ls | sort-by size | reverse
```

Shell

```
# => ╭────┬──────────────┬──────┬──────────┬──────────────╮
# => │ #  │     name     │ type │   size   │   modified   │
# => ├────┼──────────────┼──────┼──────────┼──────────────┤
# => │ 0  │ Cargo.lock   │ file │ 194.9 KiB │ 15 hours ago │
# => │ 1  │ toolkit.nu   │ file │  20.0 KiB │ 15 hours ago │
# => │ 2  │ README.md    │ file │  12.0 KiB │ 15 hours ago │
# => │ 3  │ CONTRIBUTING.md │ file │  11.0 KiB │ 5 months ago │
# => │ 4  │ ...          │ ...  │ ...       │ ...          │
# => │ 5  │ LICENSE      │ file │   1.1 KiB │ 9 months ago │
# => │ 6  │ CITATION.cff │ file │     812 B │ 2 months ago │
# => │ 7  │ Cross.toml   │ file │     666 B │ 6 months ago │
# => │ 8  │ typos.toml   │ file │     513 B │ 2 months ago │
# => ╰────┴──────────────┴──────┴──────────┴──────────────╯
```

Filtering output

Shell

```
ls | where size > 10kb
# => ╭───┬──────────────────┬──────┬──────────┬──────────────╮
# => │ # │       name       │ type │   size   │   modified   │
# => ├───┼──────────────────┼──────┼──────────┼──────────────┤
# => │ 0 │ CONTRIBUTING.md  │ file │  11.0 KiB │ 5 months ago │
# => │ 1 │ Cargo.lock       │ file │ 194.6 KiB │ 2 minutes ago │
# => │ 2 │ README.md        │ file │  12.0 KiB │ 16 hours ago │
# => │ 3 │ toolkit.nu       │ file │  20.0 KiB │ 16 hours ago │
# => ╰───┴──────────────────┴──────┴──────────┴──────────────╯
```

Processes

Shell

```
ps
# => ╭───┬──────┬──────┬──────────────┬──────────┬──────┬──────────┬─────────╮
# => │ # │ pid  │ ppid │     name     │  status  │ cpu  │   mem    │ virtual │
# => ├───┼──────┼──────┼──────────────┼──────────┼──────┼──────────┼─────────┤
# => │ 0 │    1 │    0 │ init(void)   │ Sleeping │ 0.00 │  1.2 MiB │ 2.2 MiB │
# => │ 1 │    8 │    1 │ init         │ Sleeping │ 0.00 │ 124.0 KiB│ 2.3 MiB │
# => │ 2 │ 6565 │    1 │ SessionLeader│ Sleeping │ 0.00 │ 108.0 KiB│ 2.2 MiB │
# => │ 3 │ 6566 │ 6565 │ Relay(6567)  │ Sleeping │ 0.00 │ 116.0 KiB│ 2.2 MiB │
# => │ 4 │ 6567 │ 6566 │ nu           │ Running  │ 0.00 │  28.4 MiB│ 1.1 GiB │
# => ╰───┴──────┴──────┴──────────────┴──────────┴──────┴──────────┴─────────╯
```

Running processes

Shell

```
ps | where status == Running
# => ╭───┬──────┬──────┬──────┬─────────┬──────┬──────────┬─────────╮
# => │ # │ pid  │ ppid │ name │ status  │ cpu  │   mem    │ virtual │
# => ├───┼──────┼──────┼──────┼─────────┼──────┼──────────┼─────────┤
# => │ 0 │ 6585 │ 6584 │ nu   │ Running │ 0.00 │ 31.9 MiB │ 1.2 GiB │
# => ╰───┴──────┴──────┴──────┴─────────┴──────┴──────────┴─────────╯
```

Running processes


```
ps | where status == Running
# => ╭────┬────────┬────────┬──────┬─────────┬──────┬──────────┬─────────╮
# => │ #  │  pid   │  ppid  │ name │ status  │ cpu  │   mem    │ virtual │
# => ├────┼────────┼────────┼──────┼─────────┼──────┼──────────┼─────────┤
# => │ 0  │ 6585   │ 6584   │ nu   │ Running │ 0.00 │ 31.9 MiB │ 1.2 GiB │
# => ╰────┴────────┴────────┴──────┴─────────┴──────┴──────────┴─────────╯
```

Shell

How does this work?

Running processes


```shell
ps | where status == Running
# => ╭───┬──────┬──────┬──────┬─────────┬──────┬──────────┬─────────╮
# => │ # │ pid  │ ppid │ name │ status  │ cpu  │   mem    │ virtual │
# => ├───┼──────┼──────┼──────┼─────────┼──────┼──────────┼─────────┤
# => │ 0 │ 6585 │ 6584 │ nu   │ Running │ 0.00 │ 31.9 MiB │ 1.2 GiB │
# => ╰───┴──────┴──────┴──────┴─────────┴──────┴──────────┴─────────╯
```

How does this work?

```shell
ps | describe
# => table<pid: int, ppid: int, name: string, status: string, cpu: float, mem:
filesize, virtual: filesize> (stream)
```

Running processes


```shell
ps | where status == Running
# => ╭───┬──────┬──────┬──────┬─────────┬──────┬──────────┬─────────╮
# => │ # │ pid  │ ppid │ name │ status  │ cpu  │   mem    │ virtual │
# => ├───┼──────┼──────┼──────┼─────────┼──────┼──────────┼─────────┤
# => │ 0 │ 6585 │ 6584 │ nu   │ Running │ 0.00 │ 31.9 MiB │ 1.2 GiB │
# => ╰───┴──────┴──────┴──────┴─────────┴──────┴──────────┴─────────╯
```

How does this work?

```shell
ps | describe
# => table<pid: int, ppid: int, name: string, status: string, cpu: float, mem:
filesize, virtual: filesize> (stream)
```

Exercise

Find processes sorted by greatest cpu utilization.

Exercise

Find processes sorted by greatest cpu utilization.

```shell
ps | where cpu > 0 | sort-by cpu | reverse
```

#	pid	name	cpu	mem	virtual
0	11928	nu.exe	32.12	47.7 MB	20.9 MB
1	11728	Teams.exe	10.71	53.8 MB	50.8 MB
2	21460	msedgewebview2.exe	8.43	54.0 MB	36.8 MB

Pipelines

Example


```shell
ls
| sort-by size
| reverse
| first
| get name
| cp $in ~
```

Whenever possible, Nushell commands are designed to act on pipeline input.

Why does `cp` need `$in`?

Example


```Shell
ls
| sort-by size
| reverse
| first
| get name
| cp $in ~
```

Whenever possible, Nushell commands are designed to act on pipeline input.

Why does `cp` need `$in`?

Because `cp` has two positional arguments.

No `\` needed in multi-line pipelines.

Equivalent:

```Shell
ls | sort-by size | reverse | first | get name | cp $in ~
```

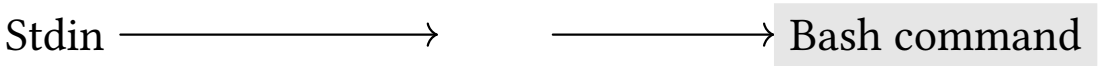
Battle of the pipelines

Bash pipeline:

```
Bash command
```

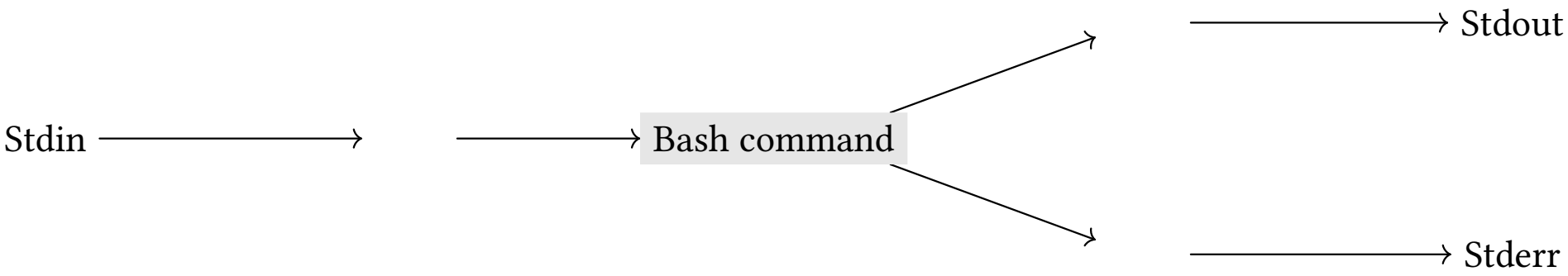
Battle of the pipelines

Bash pipeline:



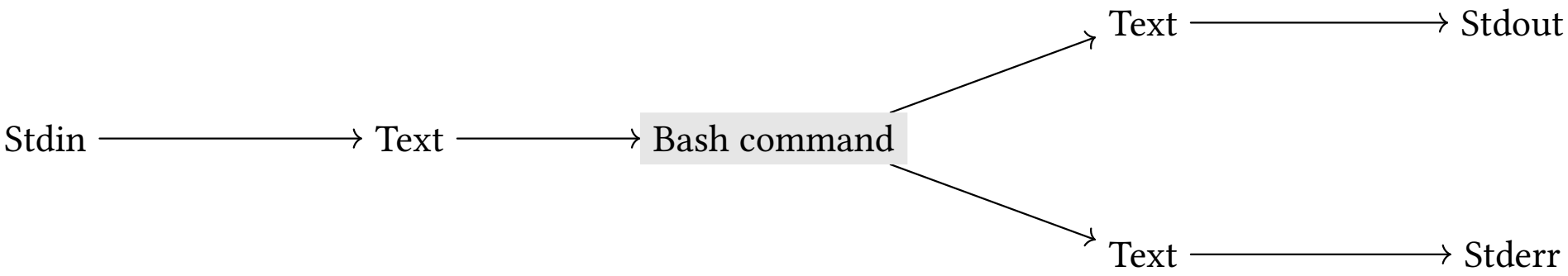
Battle of the pipelines

Bash pipeline:



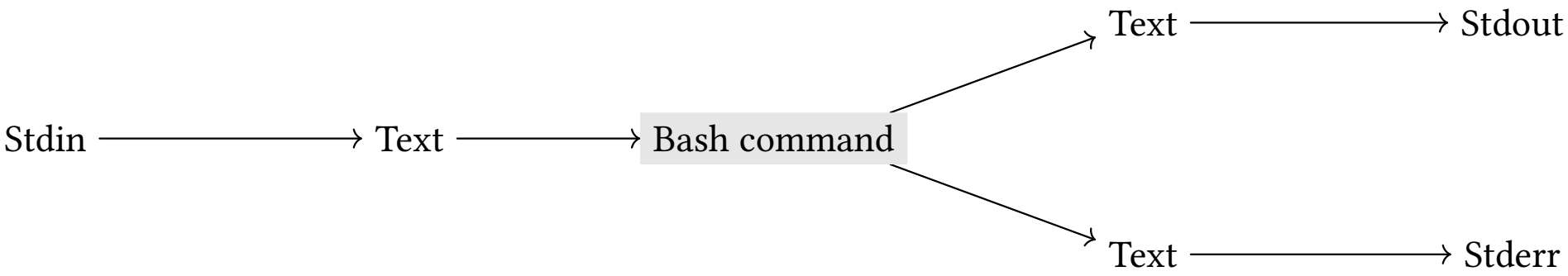
Battle of the pipelines

Bash pipeline:



Battle of the pipelines

Bash pipeline:

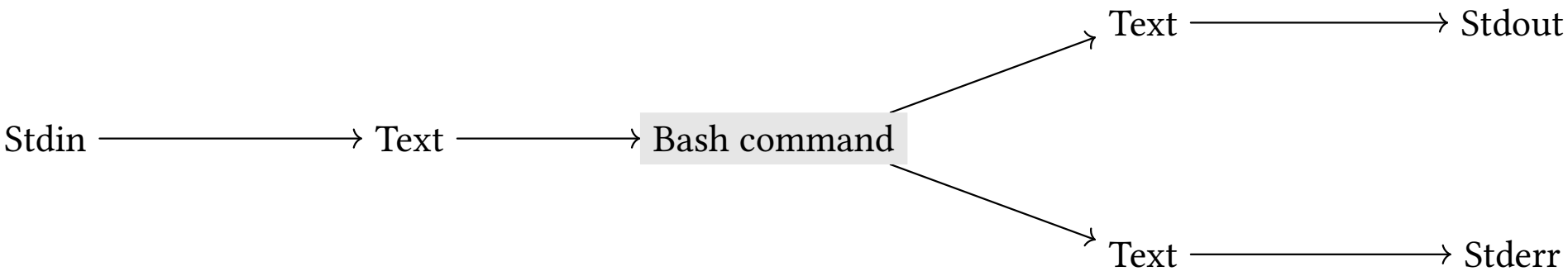


Nu pipeline:

Nu command

Battle of the pipelines

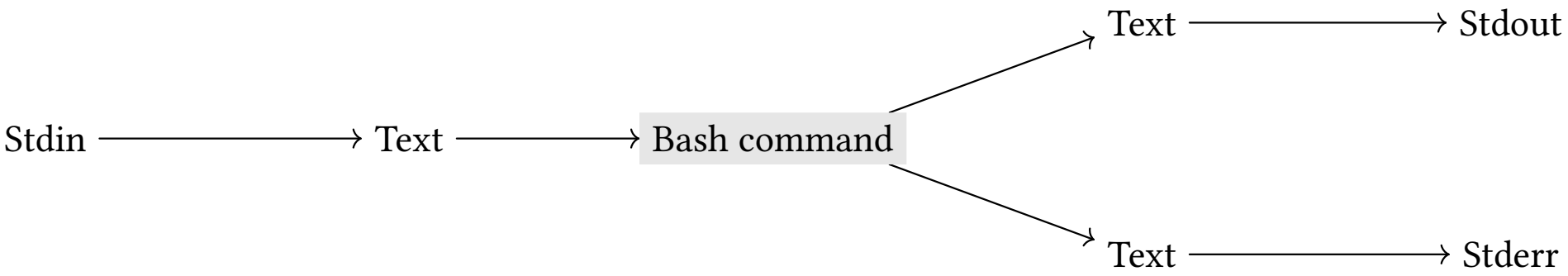
Bash pipeline:



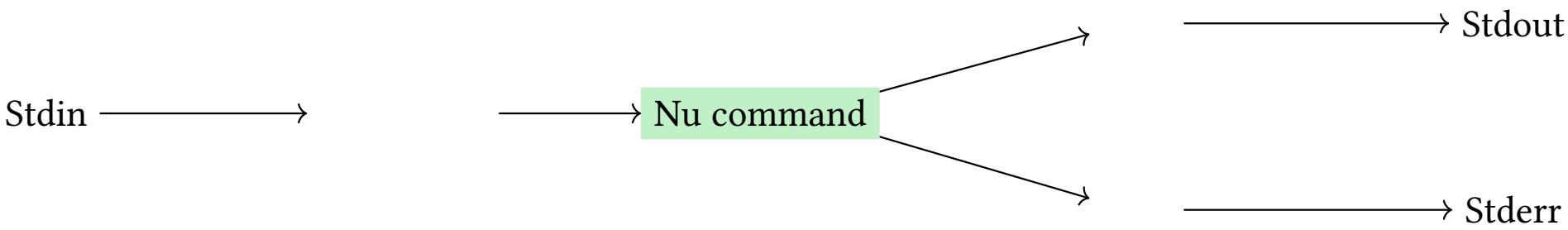
Nu pipeline:

Battle of the pipelines

Bash pipeline:

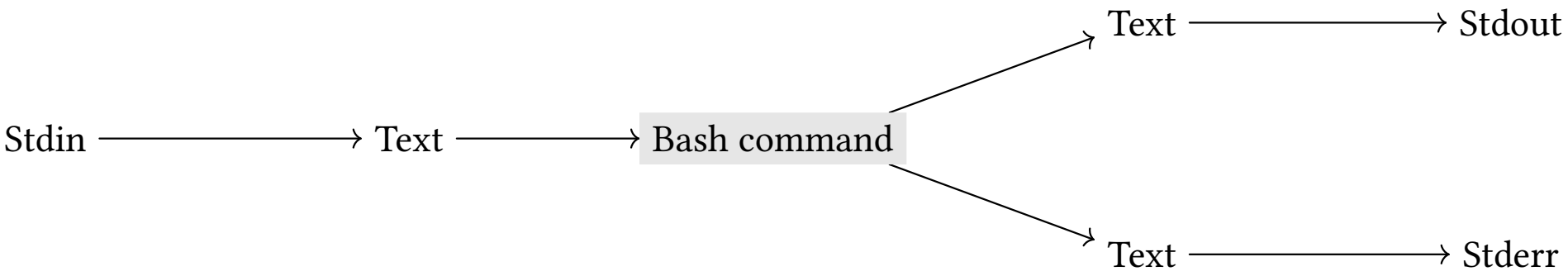


Nu pipeline:

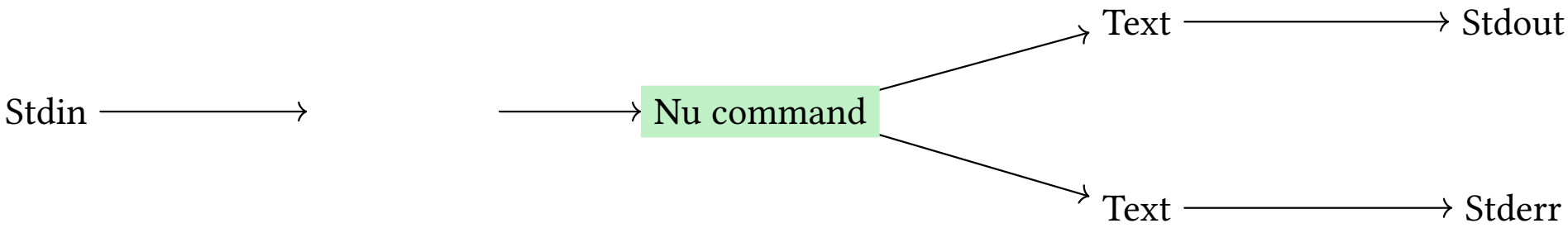


Battle of the pipelines

Bash pipeline:

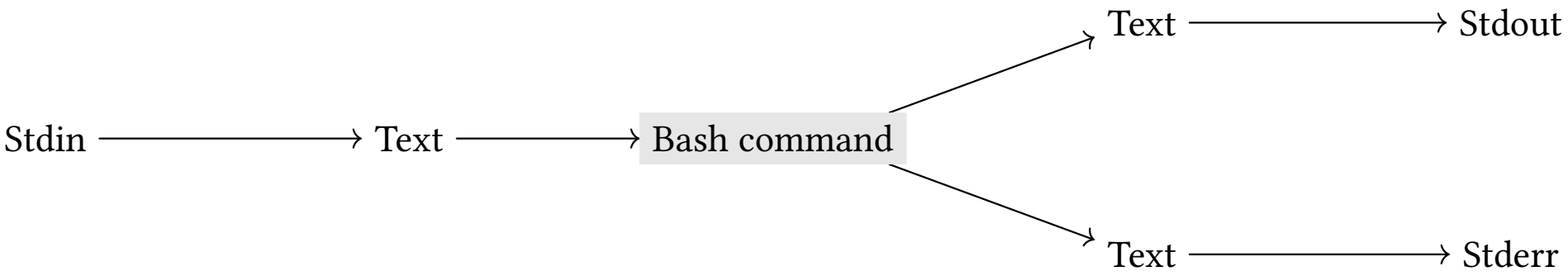


Nu pipeline:

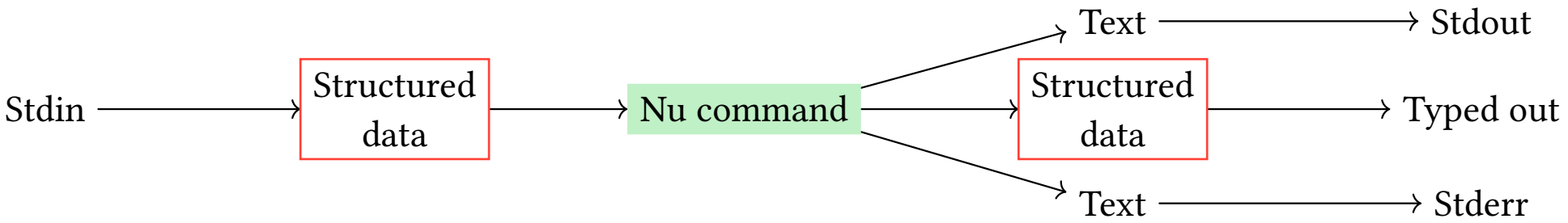


Battle of the pipelines

Bash pipeline:

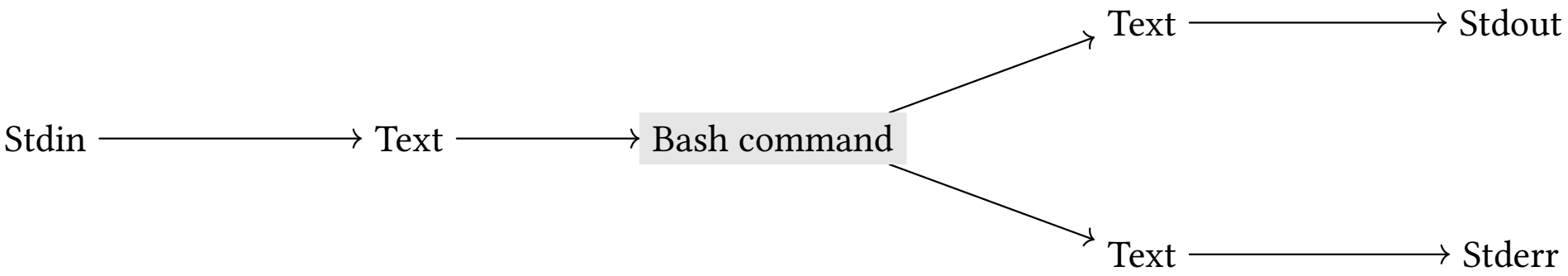


Nu pipeline:

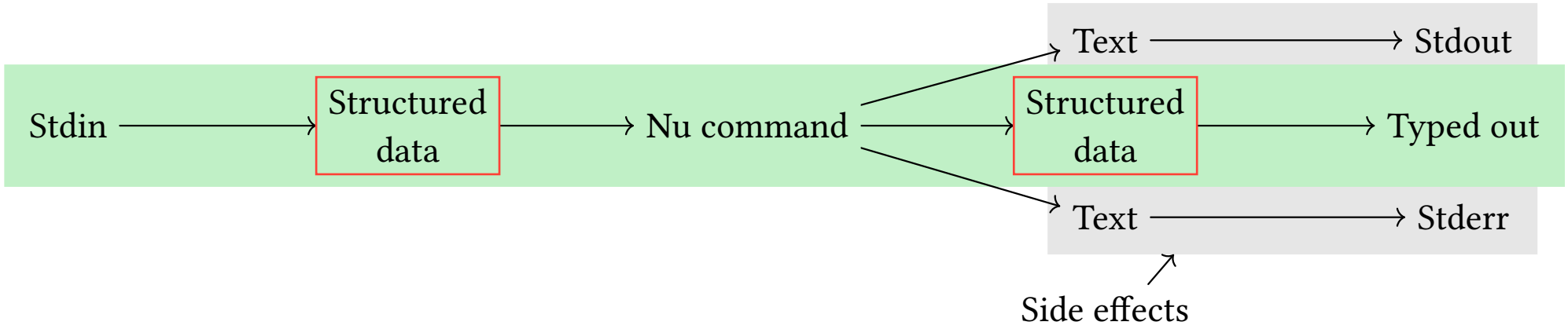


Battle of the pipelines

Bash pipeline:



Nu pipeline:



Records

Tables are built from rows of records:

```
ls                                  Shell
| sort-by size                      } (1) Table
| reverse                           }
| first                            } (2) Record
| get name                         } (3) Cell path
| cp $in ~                         } (4) String
```

Another way to find this out:

```
ls | sort-by size | reverse | first | describe                                  Shell
# => record<name: string, type: string, size: filesize, modified: datetime>
```

Exercise

Spawn a process and kill it based on its name.

Hint:

Exercise

Spawn a process and kill it based on its name.

Hint:

```shell
ps | where name == Notepad2.exe
# => ──────┬──────┬──────────────┬──────┬──────────┬─────────
# =>  #    │ pid  │     name     │ cpu  │   mem    │ virtual
# => ──────┼──────┼──────────────┼──────┼──────────┼─────────
# =>  0    │ 9268 │ Notepad2.exe │ 0.00 │ 32.0 MB  │  9.8 MB
# => ──────┴──────┴──────────────┴──────┴──────────┴─────────
```

Solution:

Exercise

Spawn a process and kill it based on its name.

Hint:

```
ps | where name == Notepad2.exe
# => ──────┬──────┬──────────────┬──────┬──────────┬─────────
# =>  #    │ pid  │     name     │ cpu  │   mem    │ virtual
# => ──────┼──────┼──────────────┼──────┼──────────┼─────────
# =>  0    │ 9268 │ Notepad2.exe │ 0.00 │ 32.0 MB  │  9.8 MB
# => ──────┴──────┴──────────────┴──────┴──────────┴─────────
```

Shell

Solution:

```
ps | where name == Notepad2.exe | get pid | get 0 | kill $in
# => ──────┬──────────────────────────────────────────────────────────────
# =>  0    │ SUCCESS: Sent termination signal to the process with PID 9268.
# => ──────┴──────────────────────────────────────────────────────────────
```

Shell

Or more concisely:

Exercise

Spawn a process and kill it based on its name.

Hint:

```
ps | where name == Notepad2.exe
# => ──────┬──────┬──────────────┬──────┬──────────┬─────────
# =>  #    │ pid  │     name     │ cpu  │   mem    │ virtual
# => ──────┼──────┼──────────────┼──────┼──────────┼─────────
# =>  0    │ 9268 │ Notepad2.exe │ 0.00 │ 32.0 MB  │  9.8 MB
# => ──────┴──────┴──────────────┴──────┴──────────┴─────────
```

Shell

Solution:

```
ps | where name == Notepad2.exe | get pid | get 0 | kill $in
# => ──────┬──────────────────────────────────────────────────────────────
# =>  0    │ SUCCESS: Sent termination signal to the process with PID 9268.
# => ──────┴──────────────────────────────────────────────────────────────
```

Shell

Or more concisely:

```
ps | where name == Notepad2.exe | get pid.0 | kill $in
```

Sh

Explore

Help

Telescoping into structured data:

```shell
help commands | explore
```

Key bindings:

- Go deeper: `Enter`
- Go back: `ESC` / `q`
- Navigate: Arrow keys or `j`/`k`

Try REPL

Open interactive data explorer with `:try` in `explore` mode.

Pipe current `explore` view into a pipeline with:

```
$in | select name description | where name == "ls"
```
Shell

(in older versions, maybe `$nu` instead of `$in`)

Exercise

Find the help page for the `cp` command and explore its output.

Use `help commands | explore` to find all commands in the `filters` category that contain "by" in their name.

Hint: In `:try` mode, use `where` and `=~` (or `str contains`).

Solution:

Exercise

Find the help page for the `cp` command and explore its output.

Use `help commands | explore` to find all commands in the `filters` category that contain "by" in their name.

Hint: In `:try` mode, use `where` and `=~` (or `str contains`).

Solution:

```
$in | where category == filters and name =~ by
```
Shell

Shorthand for:

Exercise

Find the help page for the `cp` command and explore its output.

Use `help commands | explore` to find all commands in the `filters` category that contain "by" in their name.

Hint: In `:try` mode, use `where` and `=~` (or `str contains`).

Solution:

```
$in | where category == filters and name =~ by
```
Shell

Shorthand for:

```
help commands |
where (
  ($it.category == "filters") and
  ($it.name | str contains "by")
)
```
Shell

} (1) Table output
} (2) Filter
} (3) Row condition 1
} (4) Row condition 2