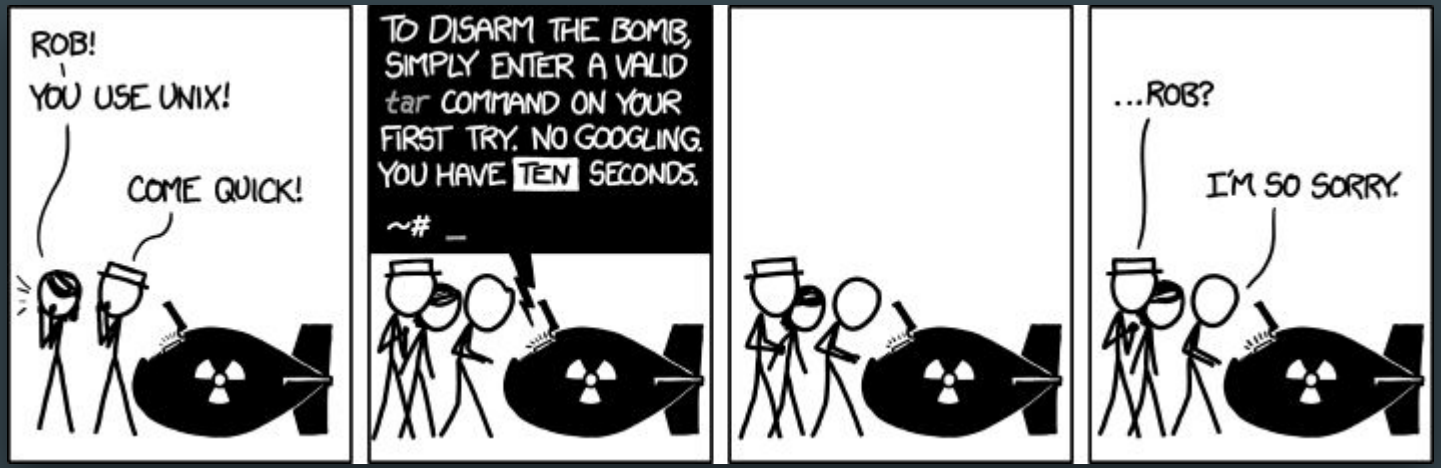
Something with CLI(s)

The Basics

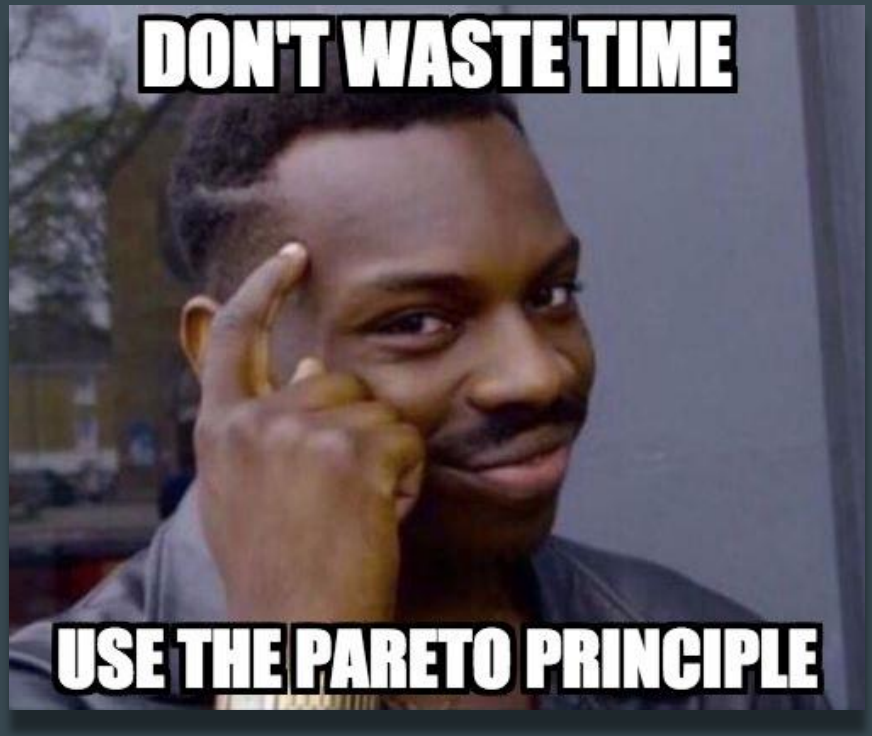
KISS



xkcd.com

80/20

focus on the essentials



1. distinguish stdout/stderr
2. return
   0 on success
   ≠ 0 on Error
3. Don’t crash with
   - Exception
   - panic
   - abort
   - ...

Configuration Layers

global to local

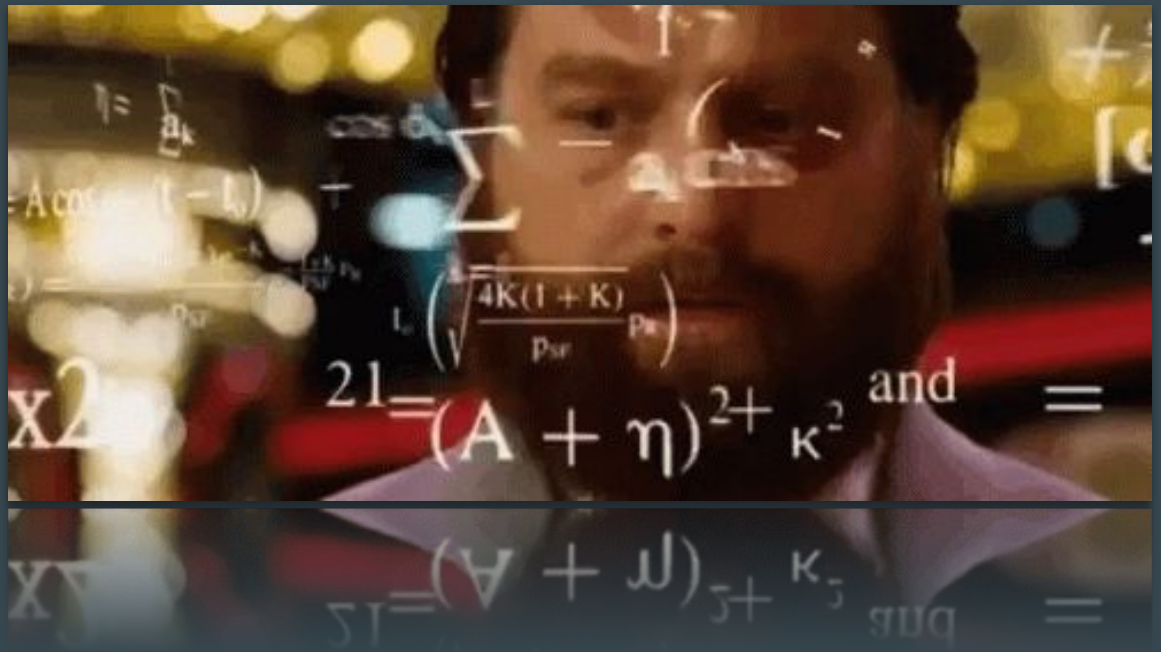
locality

- Global config
- User config
- Environment
- CLI parameters

Save yourself some programming

make use of existing CLI tools

- socat
- jq
- watch
- fzf
- tshark
- cram
- sort
- wc
- …



Examples of Cli’s

Extract data stream from udp conversation

```
user@host ~$ tshark -r $1 -Y "udp.port eq 1166 and ip.src eq ${2}" -T fields -e data | tr -d
"\n",":" | xxd -r -ps > "${2}-log.txt"
```

1. `tshark -r $1 -Y "udp.port eq 1166 and ip.src eq ${2}" -T fields -e data`
2. `tr -d "\n",":"`
3. `xxd -r -ps > "${2}-log.txt"`

Examples of Cli’s

Extract data stream from udp conversation

```
user@host ~$ tshark \
-r $1 \
-Y "udp.port eq 1166 and ip.src eq ${2}" \
-T fields -e data \
| tr -d "\n",":" \
| xxd -r -ps > "${2}-log.txt"
```

Monitor changes in json API endpoint

```
user@host ~$ watch -d  "curl http://worldclockapi.com/api/json/utc/now 2>/dev/null | jq"
```

To upper case udp/tcp echo server

```
user@host ~$ socat -ddd udp-listen:9999 system:"python3 upper.py"

user@host ~$ socat -ddd tcp-listen:9999 system:"python3 upper.py"
```

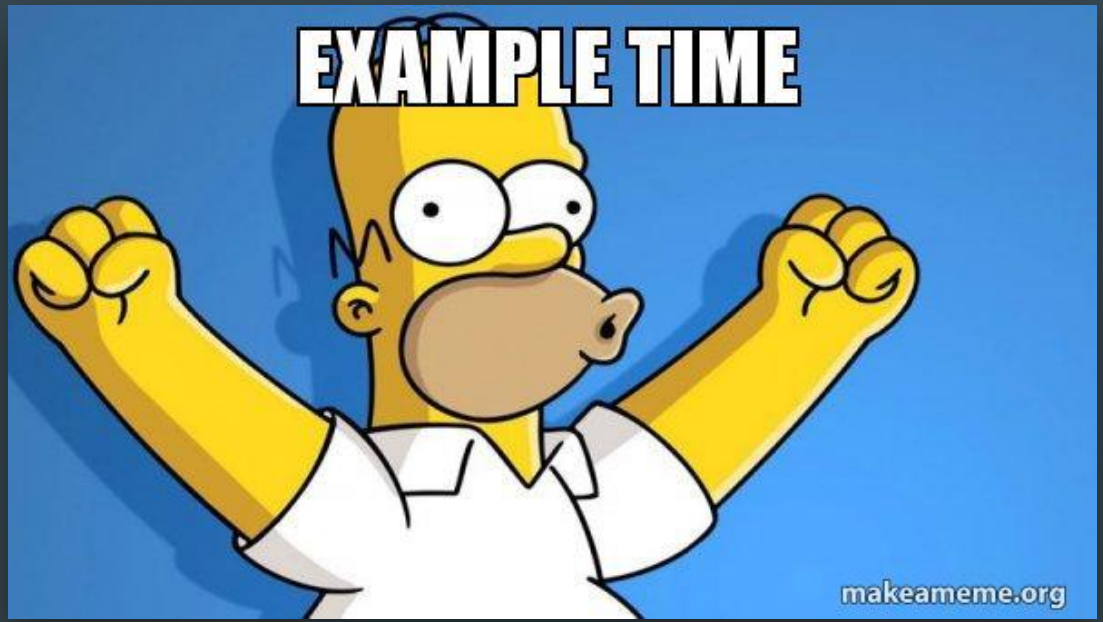
upper.py

```
import sys
for l in sys.stdin: sys.stdout.write(l.upper()); sys.stdout.flush()
```

Rust CLI Tools

real life examples

- ripgrep
- bat
- fd
- hexyl
- rogcat
- exa
- starship
- tokei
- procs
- sd



Crate support

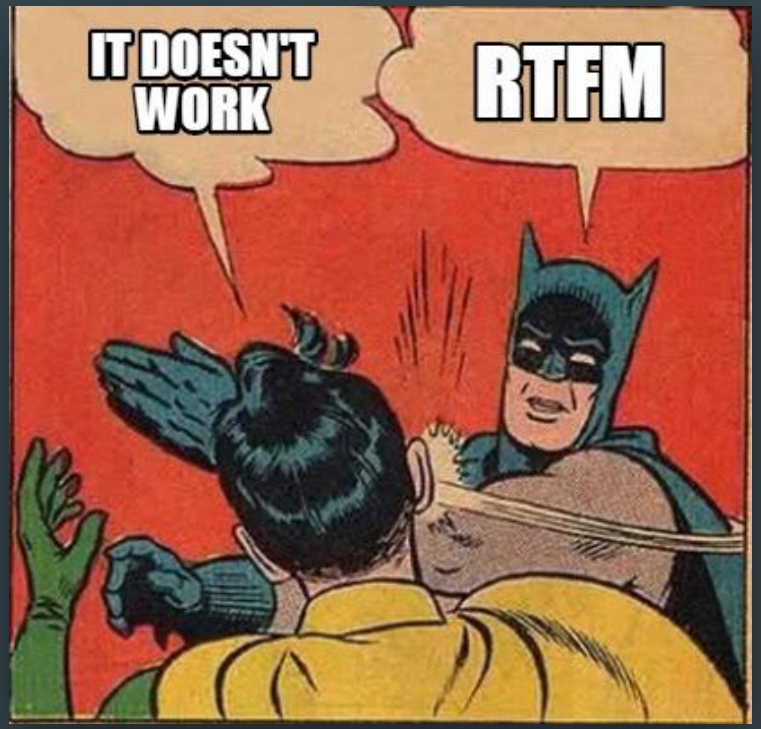
the rust toolbox

Crate	Category
clap	CLI Argument Parsing
structopt	CLI Argument Parsing
config	Configuration
anyhow	Error Handling
human panic	Error Handling
serde	Input/Output (Serialization)
log	Logging Facade
env_logger	Output (Logging)
assert_cmd	Testing
assert_file	Testing

CLI Resources

RTFM

- Rust CLI Book
- CLI Guidelines
- Socat Guide
- CLI Book
- Posix Utility Conventions



Tasks / Exercises

- Create a basic CLI application with clap/structopt
- Create a Type which defaults to stdin/stdout if - provided otherwise uses file(s) for input/output
- Support different input/output formats (text, jsonl, json)
- Write Integration test(s) for your CLI using assert_cmd, assert_file, cram
- Add configuration support via config-file/ENV to your CLI