Building a Multilingual Command Line

John Samuel

25th August, 2020

Simplify command line

Is command line

Simple?

$ ls: list files

$ touch: create a blank file

$ mkdir: create a directory

$ cd : change a directory

$ ps: list processes

$ top: list processes

$ netstat: list network connections

$ lshw: list hardware

$ lscpu : display information about cpu architecure

$ date: display or set

date

ls	: list files
touch	: create blank file
mkdir	: create a directory
cd	: change a directory
ps	: list processes
top	: list processes
netstat	: list network connections
lshw	: list hardware
lscpu	: display information about cpu architecture
date	: print or set date
...	:

Options: What about them?

`-V`: Is it verbose or is it version?

-r: Is it reverse or is it recursive?

help: Is it -h or is it --h? Is it -help?

Support from standard libraries

Standardization

Support from standard libraries

- C: getopt(), getopt_long()
- Python: argparse()

C: getopt(), getopt_long()

- short and long options
- optional arguments

Python: argparse()

- short and long options
- one or more optional arguments
- specify data types of arguments
- subcommands

Enhancement

Incorporate colors

Add progress

Multilingual documentation

2. But why not multilingual commands and options?

Multilingual Commands

- list all the processes
- affiche tous les processus
- പ്രക്രിയകൾ കാണിക്കുക

Building Multilingual Command Line: Solutions

1. Ask the developers
2. Modify the shell (e.g., Bash) source code
3. Extend the shell in a transparent user-centric manner

Inspiration from other domains

REST API

- Create C
- Read R
- Update U
- Delete D
- List L

action	resource
create	file
read	file
update	file
delete	file
list	file
(action)...	file

action	resource
créer	fichier
afficher	fichier
modifier	fichier
supprimer	fichier
lister	fichier
(action)...	fichier

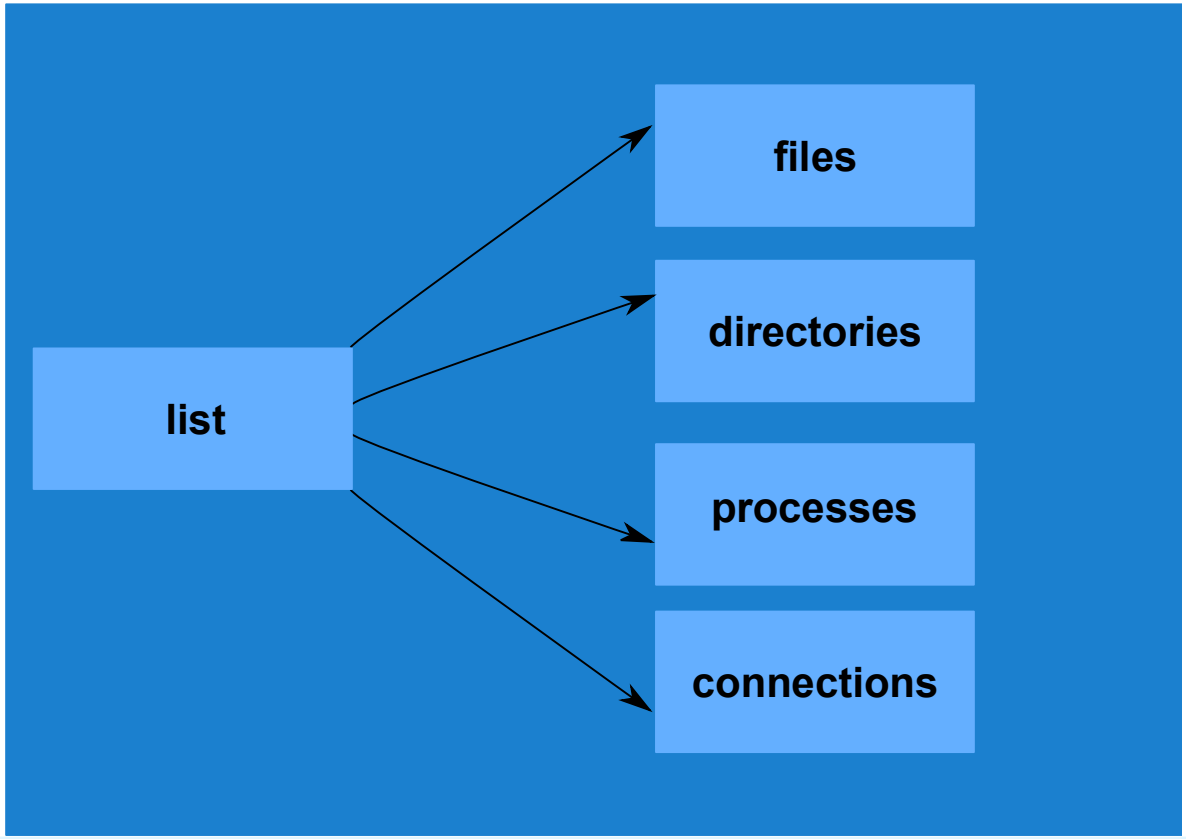

Fig 1: Command to list files, directories, processes or network connections

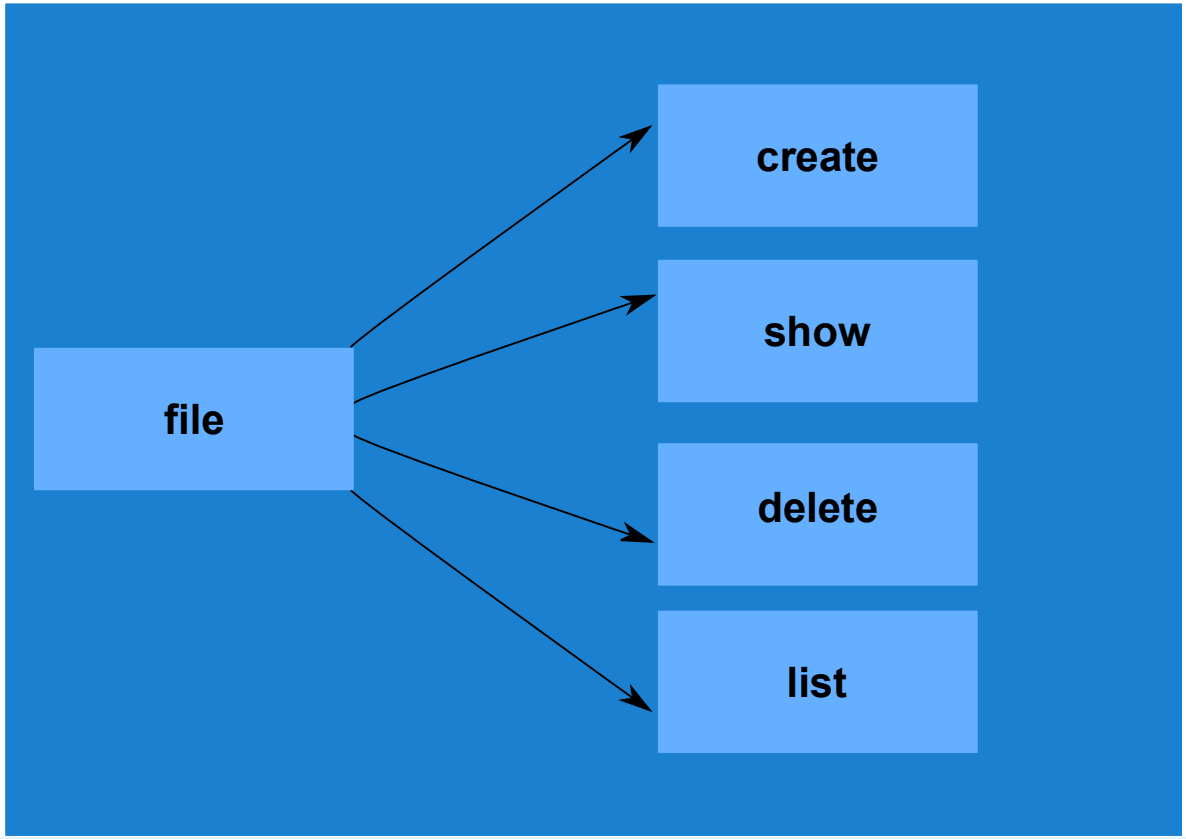

Fig 2: Command to create, show, delete and list files. Object comes in the first position.

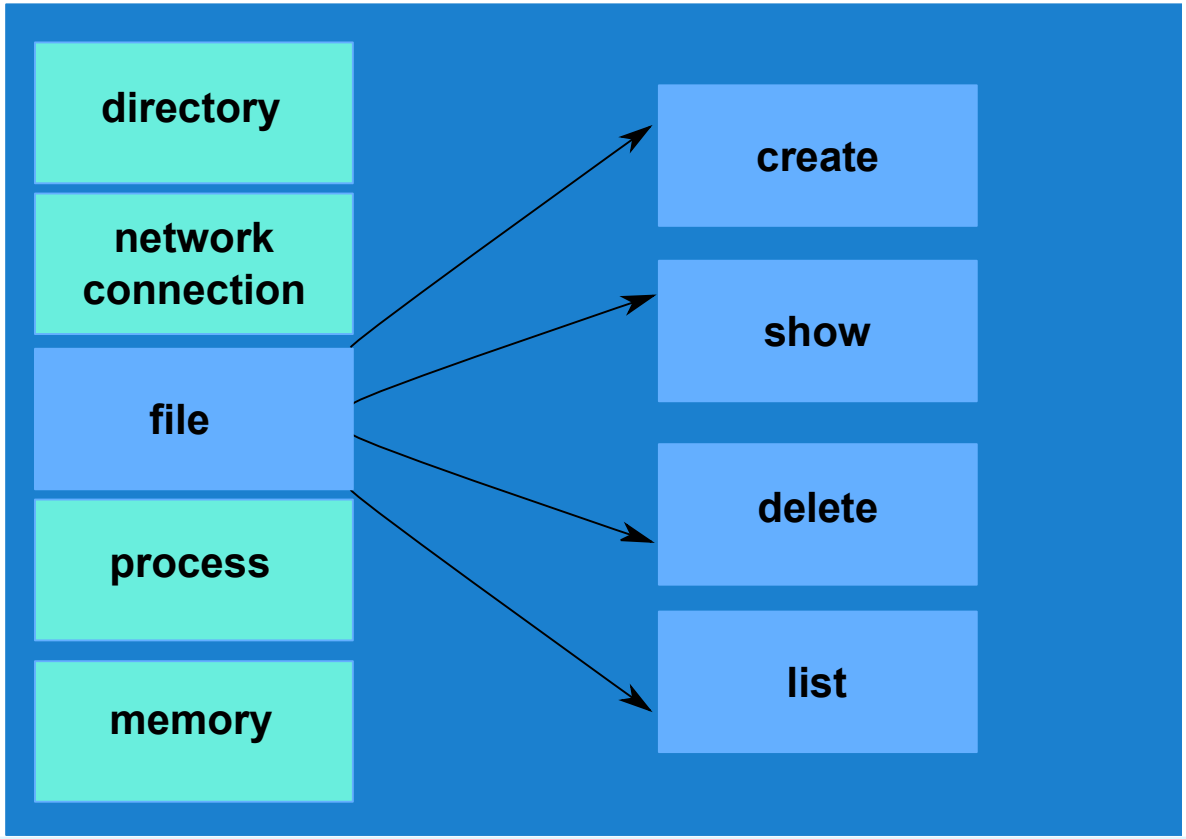

Fig 3: Command to list files, directories, processes or network connections. Object comes in the first position.

Firefox Ubiquity

Multilingual command: action
+ resource

Multilingual command:
resource + action

Multilingual command: action + resource + options

Multilingual command: resource + action + options

3. Development

3.a. Basic solution: Using Aliases

```
alias listfile="ls"
alias createfile="touch"
alias deletefile="rm"
alias showfile="cat"
```

3.a. Basic Solution: Using Aliases

```
alias listdirectory="ls"
alias createdirectory="mkdir"
alias deletedirectory="rmdir"
alias showdirectory="ls"
```

3.b. Solution: Using Functions

- create directory dir1
- show directory dir1
- delete directory dir1
- create file file1
- show file file1
- ...

3.b. Solution: Using Functions

```
function deleteaction() {
  count=$#
  if [[ $1 == "file" ]]
  then
    shift
    rm $@
  elif [[ $1 == "directory" ]]
  then
    shift
    rmdir $@
  fi
}

alias delete="deleteaction"
```

3.b. Solution: Using Functions

```
function supprimeraction() {
  count=$#
  if [[ $1 == "fichier" ]]
  then
    shift
    rm $@
  elif [[ $1 == "répertoire" ]]
  then
    shift
    rmdir $@
  fi
}

alias supprimer="supprimer"
```

3.b. Solution: Using Functions

```
function ഡയറക്ടറിപ്രവർത്തനങ്ങൾ() {
  count=$#
  if [[ $1 == "സൃഷ്ടിക്കുക" ]]
  then
    shift
    mkdir $@
  elif [[ $1 == "ഇല്ലാതാക്കുക" ]]
  then
    shift
    rmdir $@
  fi
}

alias ഡയറക്ടറി="ഡയറക്ടറിപ്രവർത്തനങ്ങൾ"
```

3.b. Solution: Using Functions

- supprimer répertoire rép1
- supprimer fichier f1

- ഡയറക്ടറി സൃഷ്ടിക്കുക ഡ1
- ഡയറക്ടറി ഇല്ലാതാക്കുക ഡ1

But, What about shorter commands and options?

3.b. Solution: Using Functions

```
function supprimeraction() {
  count=$#
  if [[ $1 == "f" ]]
  then
    shift
    rm $@
  elif [[ $1 == "r" ]]
  then
    shift
    rmdir $@
  fi
}

alias s="supprimeraction"
```

3.b. Solution: Using Functions

- s r rép1
- s f f1

4. Debian Community

Sharing and Collaboration

1. Developing multilingual command-line configuration files
2. Sharing the configuration files
3. Collaborating towards a flexible solution

$.

1. Learning curve
2. Natural Language
3. Linguistic Diversity
4. Open source

References

1. **dotfiles** https://github.com/johnsamuelwrites/dotfiles

2. **Command Line Interface** https://en.wikipedia.org/wiki/Command-line_interface

3. **Python Argparse** https://docs.python.org/3/library/argparse.html

4. **Rethinking the command line**, John Samuel, Capitole du Libre, Toulouse, France, November 19, 2017

5. **.bashrc** https://linux.die.net/man/1/bash

6. **Bash Startup Files** http://www.gnu.org/software/bash/manual/html_node/Bash-Startup-Files.html

7. **Ubiquity** https://mozillalabs.com/ubiquity/

8. **FEATURE: The linguistic command line** Aza Raskin, Interactions - Toward a model of innovation, Volume 15 Issue 1, January + February 2008, Pages 19-22

9. **Ubiquity: Designing a Multilingual Natural Language Interface** Michael Yoshitaka Erlewine, SIGIR Workshop on Information Access in a Multilingual World, July 23, 2009

Thank you

John Samuel

https://johnsamuel.info/

Questions?