Elevate your command line experience

Empowering developers with cutting-edge enhancements

Heiko Joerg Schick
Chief Architect | Industry Expert

Version 3

Today’s agenda

I. Present status

- Main requirements for the command line interface

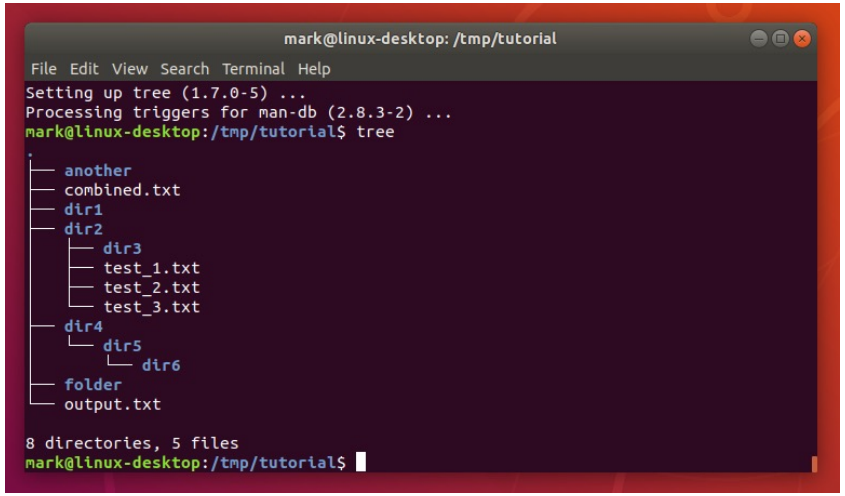
II. Result

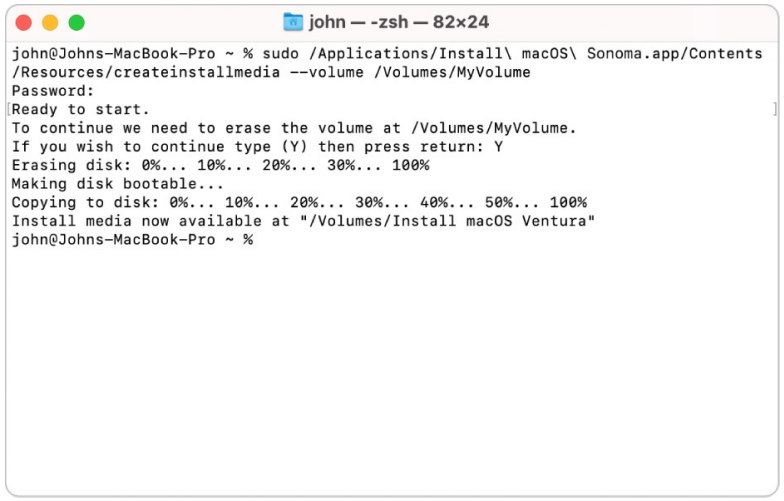
- Implementation
- Command line configuration
- Essential tools and enhancements
- Step-by-step installation and updating process
- Live demonstration

III. Conclusion and key takeaways

Present status

- In the evolving landscape of advanced technology, optimising the command line interface is crucial for efficiency and productivity. Today I detail the essential requirements, current implementations, and advanced tools that are improving command-line experiences.





- Despite advancements in computing, default command line interfaces in cutting-edge operating systems often lack intuitive features and efficient management. This results in time-consuming configurations and limited remote functionality. Additionally, ensuring consistency across various operating environments remains a significant challenge.

Main requirements for the command line interface

- Compatible with Zsh
- Operates in POSIX-compliant operating systems
- Shows the exit code of previous commands
- Displays the execution time of the previous command
- Utilises state-of-the-art tools for command history, file-system navigation, and file content display
- Proper management of configuration files (dotfiles)
- Facilitates quick changes and setups of configurations across systems
- Operates in remote computing and containerisation environments, such as SSH, VNC, and Docker
- Shows user and hostname when logged into remote environments
- Provides status indicators for tmux and Docker environments
- Shows Docker container context
- Indicates if operating as root or with sudo privileges
- Provides information on development environments for C, Python, Rust, and Zig
- Provides information on Git status and branch

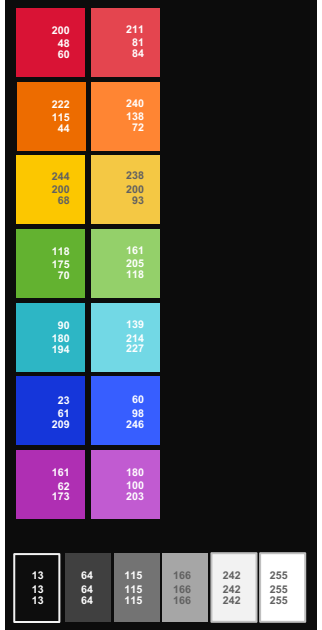
Implementation

- **Store dotfiles in a Git repository:**
  - Ensure consistency across systems by maintaining centralised dotfiles.
  - Use version control to track changes and facilitate easy updates.
- **Automate installation:**
  - Implement a Bash script to automate the installation of software prerequisites and packages.
  - Save time and reduce errors by standardising dependency installation.
- **Package management:**
  - Utilise Homebrew for streamlined software management on macOS and Linux.
  - Benefit from easy updates and access to a vast repository of tools.
- **Manage dotfiles with GNU Stow:**
  - Use GNU Stow to automate the creation and management of symbolic links.
  - Simplify the organisation and application of different configurations across environments.
- **Enhance shell experience:**
  - Combine Zsh and Starship for a more intuitive and customisable command line interface.
  - Enjoy features like auto-suggestions, syntax highlighting, and powerful theming.

Command line configuration

```
▶ ~/dotfiles                                                G(main)?
× 127 ~/dotfiles ls                                         G(main)?
▶ ~/git/rust-hello-world                    rs(v1.80.1) G(master)
▶ ~/git/helloworld-zig                      zig(v0.13.0) G(main)
▶ schihei@Archimedes ~/dotfiles             T(33s) G(main)
▶ root@schihei-laptop ~/yolo-ai-cmdbot      py(v3.10.1)env
• ▶ @9dd8880267dd ~                         D(toolchain-minimal)
• ▶ schihei@Archimedes ~/dotfiles           G(main)
```

Terminal colours (normal and bright)



Essential tools and enhancements

Name	Counterpoint	Description	Homepage
Atuin	Bash history	A powerful shell history replacement storing your shell history in an SQLite database.	Link
bat	cat	A cat clone with syntax highlighting and Git integration.	Link
delta	git diff / diff	A syntax-highlighting pager for git and diff output.	Link
eza	ls	A modern, faster alternative to `ls`.	Link
fzf	—	A general-purpose command-line fuzzy finder.	Link
Neofetch	screenfetch	A command-line system information tool that displays distro logos.	Link
pyenv & pyenv-virtualenv	virtualenv	A simple Python version management tool.	Link
Starship	—	A cross-shell prompt for astronauts. :)	Link
Stow	ln (symlink)	A symlink farm manager: a tool for managing the installation of multiple software packages.	Link
Thefuck	—	A tool that corrects errors in previous console commands.	Link
tmux	screen	A terminal multiplexer that lets you switch easily between several programs in one terminal.	Link
zoxide	cd	A smarter `cd` command, inspired by `z`, `autojump`, and `fasd`.	Link
zsh-autosuggestions	—	Fish-like fast/unobtrusive autosuggestions for Zsh.	Link
zsh-syntax-highlighting	—	Fish shell-like syntax highlighting for Zsh.	Link

Step-by-step installation and updating process

Installation:

```
$ git clone http://gitlab.h3132.de/provecta-computatione/elevate-your-command-line-experience.git dotfiles

$ cd dotfiles
$ bash install.sh
```

Update:

```
$ cd dotfiles
$ git pull
$ bash install.sh
```

Conclusion and key takeaways

- Modernising the command line interface (CLI) with tools such as Zsh, Starship, and Homebrew boosts productivity.
- Integrating POSIX standards and using Git and GNU Stow for dotfile management guarantees consistency.
- The customised command prompt provides real-time feedback and environment status.
- Automation streamlines installation and updates.
- These techniques enhance the CLI into a powerful development tool.

HJ¯S

Advanced Computing, Artificial Intelligence and Semiconductor