L'Artigianato del Terminale: Il Superpotere Segreto dello Sviluppatore Moderno

Oltre l'IDE: come la conoscenza profonda degli strumenti può fare la differenza

Come mai questo talk?

con quali sono le vostre aspettative?

Mi presento

Chi è Andrea Marisio?

Realtà

No one cares T.T

Prerequisiti: L'arsenale

Molti di questi tool sono scritti in **Rust** per le loro performance.

- **`cargo`**: Il package manager di Rust. Il modo più semplice per installare `rg`, `fd`, `bat`, `zoxide`...
- **`pyenv`** / **`pyenv-win`**: Per gestire più versioni di Python.
- **`uv`** / **`pipx`**: Per installare tool Python come `Visidata` in ambienti isolati.

La Ricerca Istantanea (7 min)

`ripgrep` (rg)

> Trovare testo. Dimenticate `grep`. `rg` è più veloce e intelligente.

Demo: confronto di velocità `grep` vs `rg`

`ripgrep`: Esempi

Trova tutte le occorrenze di "una_funzione":

```
rg "una_funzione"
```

Trova "una_funzione" solo nei file Python:

```
rg "una_funzione" -t py
```

Trova file che contengono "una_funzione":

```
rg --files-with-matches "una_funzione"
```

`repgrep` (rgr)

> `ripgrep`, ma per sostituire in modo interattivo.

```
rgr 'vecchia_funzione' 'nuova_funzione'
```

`ripgrep-all` (rga)

> `ripgrep` che cerca *dentro* a PDF, video, zip...

```
rga "testo nel pdf" .
```

`fd`

> Trovare file. Dimenticate `find`. `fd` ha una sintassi umana.

Demo: confronto di sintassi `find` vs `fd`

`fd`: Esempi

Trova tutti i file che finiscono con `.md`:

```
fd -e md
```

Trova tutti i file che iniziano con `main`:

```
fd "^main"
```

Esegui un comando su ogni file trovato:

```
fd -e py --exec chmod +x
```

La Visualizzazione Intelligente (7 min)

`bat`

> Visualizzare file. Dimenticate `cat`. `bat` vi dà syntax highlighting e integrazione con Git.

Demo: Mostra un file sorgente, evidenziando le modifiche Git.

`bat`: Esempi

Visualizza un file con syntax highlighting e numeri di riga:

```
bat main.py
```

Mostra solo le righe dalla 10 alla 20:

```
bat -r 10:20 main.py
```

Integrazione con `git diff`:

```
git diff | bat
```

`zoxide`

> Navigare directory. Dimenticate `cd ../..`. `zoxide` impara e vi teletrasporta.

Si usa con un alias, solitamente `z`.

Demo "magica" di `z`.

`zoxide`: Esempi

Vai a una directory che contiene "project":

```
z project
```

Vai a una sottodirectory:

```
z my-project/src
```

Mostra le directory più usate e scegli interattivamente:

```
zi
```

Il Legante Magico: L'Interattività (8 min)

`fzf`

> E ora, leghiamo tutto insieme. `fzf` può prendere l'output di qualsiasi comando e trasformarlo in un menu interattivo.

LA KILLER COMBO

> "Voglio modificare un file, ma ricordo solo una funzione al suo interno."

```
nvim $(rg --files-with-matches "nome_funzione" | fzf)
```

1. `rg` trova tutti i file che contengono la funzione.
2. `fzf` mi fa scegliere quale file voglio da una lista interattiva.
3. `nvim` lo apre.

Questo è il momento in cui la gente prende appunti e pensa "Ok, questo mi cambia la vita".

`tldr`

> "Come si usava `tar`?"

- **`tldr`**: Manuali (`man`) semplificati con esempi pratici.

```
# Esempio di vita reale
tldr tar
```

Bonus: Gestire Git senza Sforzo (4 min)

`lazygit`

> Infine, per Git. Invece di memorizzare decine di comandi, potete avere un'interfaccia TUI potentissima.

Demo: stage, commit e log in 30 secondi.

`Visidata`

> Un coltellino svizzero per i dati nel terminale. Esplora, modifica e analizza CSV, JSON, Excel e molto altro.

```
vd data.csv
```

`tmux` & `zellij`

> Terminal Multiplexers: sessioni, finestre e pannelli persistenti. Il vostro lavoro sopravvive alla chiusura del terminale.

- `tmux new -s progetto` : Avvia una nuova sessione `tmux` nominata.
- `zellij` : Avvia `zellij` (spesso con una configurazione più intuitiva).
- `<Ctrl-b> d` : (in tmux) Detach, lascia la sessione in background.

`jq` & `yq`

> Il `sed` del mondo moderno per JSON e YAML. Filtrare, mappare e trasformare dati strutturati da CLI.

```
# Estrai un valore da un JSON
cat file.json | jq '.primo_livello.secondo[0]'

# Modifica un file YAML al volo
yq -i '.chiave.sottochiave = "nuovo_valore"' config.yaml
```

`fish` & `zsh`

> Shell moderne che potenziano l'interattività.

- `fish` : Completamenti, suggerimenti e sintassi colorata da subito.
- `zsh` + `oh-my-zsh` / `prezto` : Un ecosistema enorme di plugin e temi per personalizzare ogni aspetto.

Conclusione (2 min)

Conclusione

Questi non sono tool a caso. Sono i pezzi di un puzzle che, una volta assemblato, crea un workflow incredibilmente potente.

Vi invito a provarne anche solo uno.

Risorse

Trovate slide, configurazioni e link su:

```
git@github.com:AM-I-Human/am-i-a-terminal-artisan.git
```

Grazie!