**Extremely Cool Command Line Tools - Supercharged with Rust - {Part 1}**

Skilled network engineers, developers, system admins, etc. use Unix/Linux commands line tools/utilities such as …. **cat**, **ls**, **man**, **ps**, etc..

Most of these were originally written for Unix or Unix-like operating systems, have been around for decades and, while they are still very useful, we can now take them to the next level with their equivalents in the Rust language. Some are upgrades to existing tools and some are brand new and much needed.

Rust is a newer language (2010) that has become extremely popular. It is a powerful, fast, multi-platform, multi-paradigm, high-level, general-purpose programming language, which was recently added to the Linux kernel. According to the Stack Overflow Developer Surveys, Rust has been the "most loved programming language" every year from 2016 to 2022 (inclusive).

Just install Rust and each tool you want to try out. Most of these tools can be installed the standard operating system package managers, or you can use the Rust package manager, known as Cargo, which should work on most platforms. Here are the command line tools I’ll be talking about in this article:

**Atuin** - magical shell history

**bat** - A **cat**(1) clone with wings

**difftastic** - A structural **diff** that understands syntax

**lsd** - An **ls** command with a lot of pretty colors and some other stuff

**names** - A random name generator for use in container instances, project names, etc

**procs** - A modern replacement for **ps**

**ripgrep** - supports many of the same features found in **grep**, but is super fast

**tealdeer** - Fetch and show tldr help pages for many CLI commands.

If you need help installing and setting up Rust, just scroll to the bottom of this article.

Let the excitement begin!

**Atuin** - magical shell history

A picture containing logo

Description automatically generated

Follow install instructions:

<https://lib.rs/crates/atuin>

Tabbing-up to find previous commands can often make things much easier in your Terminal, Shell, IDE, CLI, etc.

Atuin takes that concept to the next level. Once, installed, tabbing up will give you this cool GUI experience, showing all recent command and when they were executed.

Text

Description automatically generated

Just tab up like you normally would and hit ENTER when the command you want is highlighted.

Atuin replaces your existing shell history with an SQLite database and records additional context for your commands. Additionally, it provides optional and *fully encrypted* synchronization of your history between machines, via an Atuin server.

**bat** - A **cat** clone with wings



Follow install instructions:

<https://lib.rs/crates/bat>

The **cat** command in Linux has long been the one of the primary tools for viewing the contents of text files. It gets the job done, but it’s from 1971 and, well, basic.

**bat** supports display of colorful, customizable syntax highlights, plus line numbers, git modifications, non-printable characters and more. For me, the choice is clear:

**cat**

Text

Description automatically generated

**bat**

Text

Description automatically generated

**difftastic** - A structural **diff** that understands syntax



Follow install instructions:

<https://lib.rs/crates/difftastic>

Back in the 1970s, the **diff** tool was created to compare files and shows their differences.

Let’s compare two files I created, using both **diff** and **difftastic**:

**diff** my\_code\_before.py my\_code\_after.py

Text

Description automatically generated

**difft** my\_code\_before.py my\_code\_after.py

Text

Description automatically generated

I’ll let you decide which one is more helpful.

**lsd** - An **ls** command with a lot of pretty colors and some other stuff

**A picture containing icon

Description automatically generated**

Follow install instructions:

<https://lib.rs/crates/lsd>

**ls** has been around since 1987 and it lists files and directories. **lsd** is a rewrite of GNU **ls** with a lot of added features, such as colors, icons, tree-view, and more formatting options.

**ls**

A picture containing calendar

Description automatically generated

**lsd**

A picture containing graphical user interface

Description automatically generated

Ah, much better!

**names** - A random name generator with names suitable for use in container instances, project names, application instances, etc.

****

Follow install instructions:

<https://lib.rs/crates/names>

**# 5 random names**

alexstev % **names** 5

highfalutin-fireman

mindless-lift

chilly-coat

berserk-interest

abnormal-bag

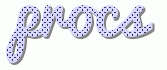
**# 2 random names with a random 4-number pad**

alexstev % **names** –number 2

equal-writing-7043

shivering-partner-8412

**procs** - A modern replacement for **ps**



Follow install instructions:

<https://lib.rs/crates/procs>

**ps** has been used to show “process status” on Unix-like systems since the dawn of time (ok, maybe not that long ago, but it has been 50 years, so close ; ).

**procs** includes colored and human-readable output, automatic theme detection based on terminal background, multi-column keyword search, tree view and more.

Here’s what **procs** has, that **ps** doesn’t:

* TCP/UDP port
* Read/Write throughput
* Docker container name
* More memory information

Let’s see the difference between inspecting process id# 182 with **ps** versus with **procs**

**ps** 182



**procs** 182



Platform

* Linux is supported.
* macOS is experimentally supported.
  + macOS version is checked on Travis CI environment only.
  + The issues caused by real-machine are welcome.
* Windows is experimentally supported.
  + Paging is not supported

**ripgrep** - a line-oriented search tool that recursively searches the current directory for a regex pattern, while respecting gitignore rules and automatically skip hidden files/directories and binary files

Follow install instructions:

<https://crates.io/crates/ripgrep>



**ripgrep** supports many of the same features found in **grep**, such as showing the context of search results, searching multiple patterns, highlighting matches with color and full Unicode support. Unlike GNU **grep**, **ripgrep** stays very fast while supporting Unicode (which is always on), and even searches compressed files!

Let’s compare the results of grep with the results of ripgrep

**grep** "json" parsing\_json.py

Text

Description automatically generated

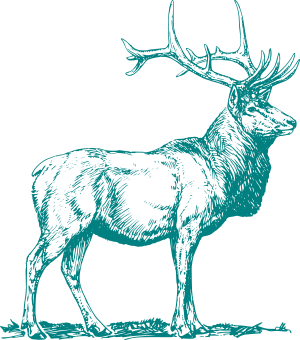
**rg** "json" parsing\_json.py

Text

Description automatically generated

I’m not saying rest in peace (r.i.p.) to **grep**, because there are times to still use grep, such as if you need a portable tool which conforms to any standard, such as POSIX; while ripgrep works on Windows, macOS and Linux, it is not ubiquitous.

**tealdeer** - Fetch and show tldr help pages for many CLI commands.



Follow install instructions:

<https://lib.rs/crates/tealdeer>

**teeldeer** is an extremely fast implementation of [tldr](https://github.com/tldr-pages/tldr) in Rust; it provides simplified, example based and community-driven **man** pages. TLDR is internet slang for “Too Long Didn’t Read” and the original **tldr** tool was designed more recently (within the last 5 years, I believe) for people who didn’t want or have time to read the entire **man** page of a tool.

Let’s look at the difference between the outputs of “man ls” and “tldr ls”.

**man** ls

(no matter how much I shrink it, it won’t fit on one page)

Text

Description automatically generated

**tldr** ls

Text

Description automatically generated

Simply beautiful!

**Installing and Setting Up Rust**

1. Rust is installed and managed by the **rustup** tool. Add it to your PATH by appending one of the following to your .bash\_profile, .bashrc or .zshrc configuration files:

export PATH="$HOME/.cargo/bin:$PATH"

1. You can install Rust using cURL. The most recent version of the command will be found at <https://www.rust-lang.org/tools/install>

At the time of writing, the command looks like this:

curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

1. Restart your shell with ‘exec bash’, ‘exec zsh’, etc..
2. Check that Rust is installed:

rustc –version

rustc 1.67.1 (d5a82bbd2 2023-02-07)

We’re good to go!

If you don’t want to learn the names of the new commands, you can create aliases and map the new tools to the old names. For instructions of how to do that, see my Knowledge Base Article titled ‘[**Developer Basics: Creating Aliases with bashrc and zshrc**](https://community.cisco.com/t5/developer-general-knowledge-base/developer-basics-creating-aliases-with-bashrc-and-zshrc/ta-p/4672529)’

This was just the first installment in this series; more articles to come!

Inspiration for this article was provided by DJ Ware [@CyberGizmo](https://www.youtube.com/@CyberGizmo)