

APPS@UCU

Linux course

Tools overview

Morhunenko Mykola



Contents

- 1 Shell
- 2 Vim
- 3 Tmux
- 4 Ranger
- 5 i3
- 6 Package managers
- 7 Sources

- In this presentation, we will overview some tools that are available on all Linux distributions
- All of them have a high barrier to entry as Linux itself, but when you are there - you won't imagine your life without that tools
- Example: it's not a one-day task to learn how to move around your system, but after few months of practice working with CLI, GUI for you will be as slow as a turtle is slow in comparison with a rabbit

All you need is

`#!/bin/bash`

The most important tool

- |



- So let's start with the text editor
- You have heard about **vim** , haven't you?
- But let's start with **vi**
- Vi is a part of POSIX
- It's totally CLI editor (forget that you have a mouse)
- There are shortcuts for everything
- If there is no, you can create them for yourself
- Every good enough 21'st century editor has an extension for a **vi mode**
- But almost nobody uses it - only on some low memory and low power machines. So we move to vim



- **Vim** stands for 'Vi IMproved'
- According to Linux Journal survey, 38% (in average for 2009-2018) of respondents vote for vim as the best editor
- It has much more features, than **vi**, including more commands, scriptable syntax highlighting and extensions, graphical interface (and a mouse support, but don't use it)
- As **vi**, it has six modes - normal, visual, insert, command-line, select, and ex (yes, not only NORMAL and INSERT)
- Because of a huge community (38% of world's best geeks) **vim** became a powerful IDE with thousands of extensions (syntax highlight, autocompletion, spell checking, project tree etc)
- The most powerful tool of **vim** is inside - its shortcuts. You can make your work dozens of times faster without a touchpad and a mouse

- Neovim is just a fork of Vim with some Python extensions
- And cool logo =)
- Also Neovim is a community-driven text editor, while Vim is a project of only one person - [Bram Moolenaar](#)
- One 'expert' on reddit wrote that:
"Neovim exists to convince Bram to push new features to Vim"
And I mostly agree with him.

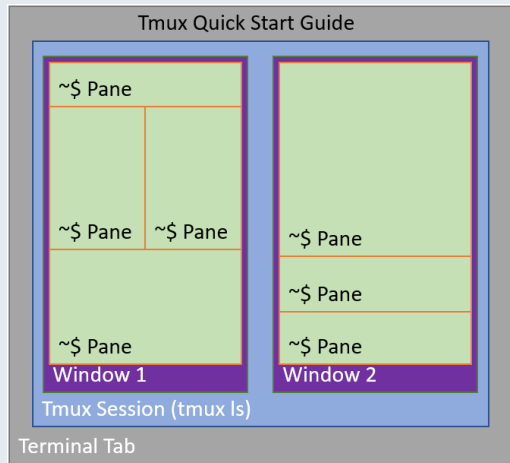


Tmux

- **TMUX** stands for Terminal MULTipleXer
- There are some other (screen, Konsole, etc.), but they are not so good as TMUX
- Why do we need it?
- As you continue your practice in CLI, you can notice that it's not enough to have only one terminal window
- With this much multitasking going on, we want to have more terminals. So people create a **terminal multiplexor**
- What TMUX can do?
 - Not only split and stack tab but also make tabs
 - Continue running programs in the background
 - With extensions you can write layout files in **.yaml** format
 - Search through terminal output and move around with Vim shortcuts
 - Other interesting stuff

Tmux

- A Tmux Session with two tmux tabs with multiple tmux panes within each
- As `vim`, tmux has modes - view and command (Ctrl+b be default)
- Every pane has three modes - view, choose and copy
- To enter a `copy mode` - `Ctrl+b [`
- It allows you to use vim keys for moving around and copying text
- For more information see `man tmux` or `linux man page`



For the very beginning:

[tmux sessions]		linuxacademy.local		[tmux windows]		linuxacademy.local	
_ new sessions tmux tmux new tmux new-session tmux new -s sessionname		_ remove sessions tmux kill-ses tmux kill-session -t sessionname		_ windows are like tabs in a browser. Windows exist in sessions and occupy the space of a session screen.		Ctrl + B 0 .. 9 select window by number	
_ attach sessions tmux a tmux att tmux attach tmux attach-session tmux a -t sessionname		_ key bindings Ctrl + B \$ rename session Ctrl + B D detach session Ctrl + B) next session Ctrl + B (previous session		_ key bindings Ctrl + B C create window Ctrl + B N move to next window Ctrl + B P move to previous window Ctrl + B L move to window last used		Ctrl + B ' select window by name Ctrl + B . change window number Ctrl + B , rename window Ctrl + B F search windows Ctrl + B & kill window	
[tmux panes]		linuxacademy.local		[tmux copy mode]		linuxacademy.com	
_ panes are sections of windows that have been split into different screens - just like the panes of a real window!		Ctrl + B ↑ move up to pane Ctrl + B ↓ move down to pane Ctrl + B O go to next pane Ctrl + B ; go to last active pane		_ key bindings Ctrl + B [enter copy mode Ctrl + B] paste from buffer		G go to bottom h move cursor left j move cursor down k move cursor up l move cursor right / search # list paste buffers q quit	
_ key bindings Ctrl + B % vertical split Ctrl + B " horizontal split Ctrl + B → move to pane to the right Ctrl + B ← move to pane to the left		Ctrl + B } move pane right Ctrl + B { move pane left Ctrl + B ! convert pane to window Ctrl + B X kill pane		_ copy mode commands space start selection enter copy selection Esc clear selection g go to top			



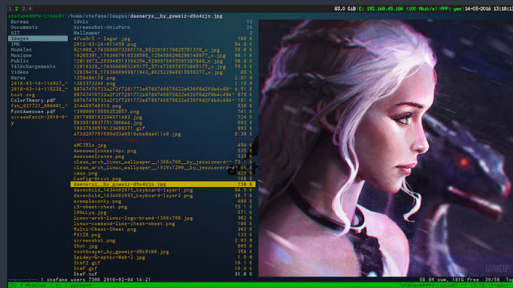
Commander

- Norton - one of the very first **dual pane file managers**, 1984
- Norton Commander set the tone for decades of file managers (Commanders) to come
- Until then people created nothing better than that, so DP commanders are still popular
- **mc** (midnight commander)
- **dc** (double commander)
- There are a lot of both GUI and CLI examples, for Linux and Windows, but we will view cpecific one - **ranger**



Ranger

- Ranger is **vim** inspired CLI file manager, so it has **vi** keybindings
- It is fully customisable with just few files
- As you can see, it can open images preview right in the terminal
- The same about all text files, videos, other files too
- For more info see **man ranger**



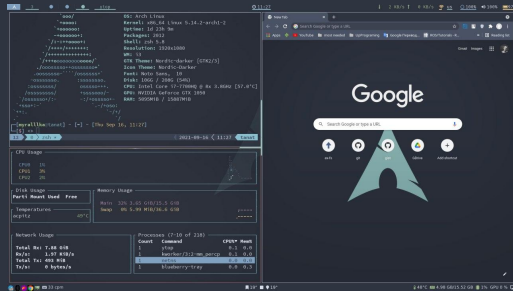
i3wm



- We will talk more about graphics in one of the following lectures
- Short explain: there are three main items in every GUI on your pc:
 - DM - Display manager
 - WM - Window manager
 - DE - Desktop environment
- As you can see, **i3wm** is a window manager
- It is quite similar to everything above in this lecture: vim shortcuts and tmux approach (but for GUI applications)
- In **i3** we also have windows and panes, but also **workspaces**
- All settings located in one file - **/.config/i3/config**
- As far as there is no DE for i3, you should install everything for yourself (all applets and programs, top/bottom bar, menu)

i3wm example

- Here you can see browser and two terminal emulators opened
- Also **polybar** with it's applets used as a top and bottom bars
- **dmenu** used as a menu
- All windows located on a same **z** level, new windows move previous one so all opened windows are visible (as in **TMUX**)
- It is called **tiling window manager** and I recommend it because it is simpler and faster, then all other types
- For more info you can see **man i3**



Package managers

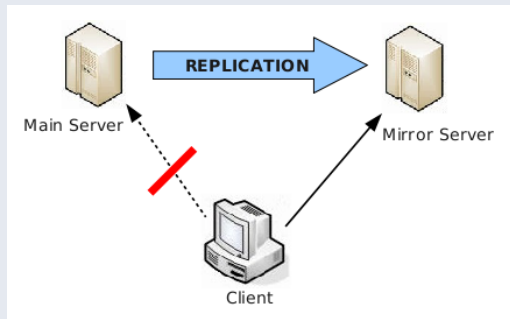


Package managers

- This is last but not least topic for today
- Any OS is just a batch of programs
- And Package manager is a tool for installing those programs (also upgrading, configuring, removing, resolving dependencies)
- There are few core new concepts, that we need to understand how it works

Mirrors

- **Mirror server (mirror)** - servers that located (physically) in different locations, but contain the same data
- For example: You want to install something, but your internet connection is too slow (or something happened to server at the US), so you download package for installation from server located in Germany (or other location)
- See your current list of mirrors (Arch-based): `vim /etc/pacman.d/mirrorlist`
- Sort them by speed: `sudo fetchmirrors -c UA`

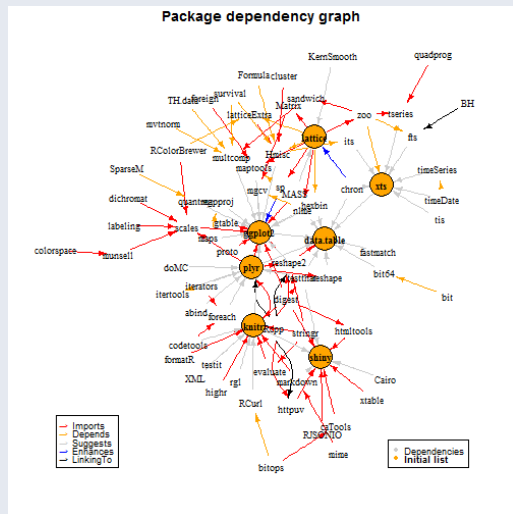


Dependencies

- Another core concept called **dependency**
- When you write any program, you use some existing libraries. So your program **depend on** that libraries/tools
- All that libraries have its versions
- As far as people change stuff in their programs, API can be different from one to other version
- Good package managers can resolve all that dependencies easily (**debian's apt can't**)
- There are **immediat** (your program use it) and **transitive** (your dependencies use it) dependencies

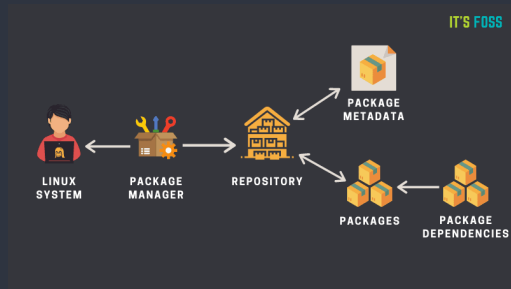
Dependencies

- A lot of programming languages has it's own package managers (**pip** for Python, **cargo** for Rust)
- Almost every Linux-based OS has it's own package manager
- All packages are **archives** with **program** itself and **metadata** - software's name, description of its purpose, version number, vendor, checksum, list of dependencies



Repositories

- One more important thing - repo
- There are official and side repositories
- Package manager supports Official repos by default, but to use some side repos, you should add it manually
- Different PacManagers have different approaches for managing repos
- **Apt** repos are defined in `/etc/apt/sources.list` and in `/etc/apt/sources.list.d` directory



Pacman

- Why we love Arch Linux so much?
Mostly because of this guy =)
- Pacman has all settings in
`/etc/pacman.conf` file
- Main repositories are:
 - `core` (main OS elements)
 - `extra` (not main, but important OS elements as GUI)
 - `community` (packages that have been adopted by Trusted Users from AUR)
 - `multilib` (32-bit software and libraries for old stuff)
 - `AUR` (Arch User Repository, the best bigger Linux repo)



Releases

- **Release system** is the concept of frequently delivering updates to applications
- It usually depend on the OS (but also on PacManagers)
- **Rolling Release** . There is no such thing as **Arch 1** , **Gentoo 2** or **Void Linux 3** . Thats because these OS's have Rolling Release model - it's repos contain all (not always stable) new programs, but also previous versions of all packages. Good for people who like test new programs or languages, real geeks. Often updates require
- **Stable Release** . Properly tested version of the product is released, sometimes half a year after it first appears. For example, **python 3.9** was in **pacman** two days after official release, but in **apt** - more then half a year after. Good for average users, easy to maintain (updates are only every week/month)
- **LTS** - Long term support. For this system package manager **freeze** some special versions of all libraries and programs, and only **minor bug or security fixes** are released for a really long term (sometimes up to decades). Good for corporations, big companies. If something is not working - just reinstall the system as it was at the very beginning, and everything will work again. The easiest to maintain

Sources

Sources

- [Linux journal](#)
- [Termianl Multiplexers](#)
- [Tmux tutorial](#)
- [Tmux Linux man page](#)
- [Dual pane file manager history](#)
- [Ranger github page](#)
- [Wiki package managers](#)
- [Pacman ArchWiki](#)
- [External Repositories Ubuntu](#)
- [Releases Wiki](#)