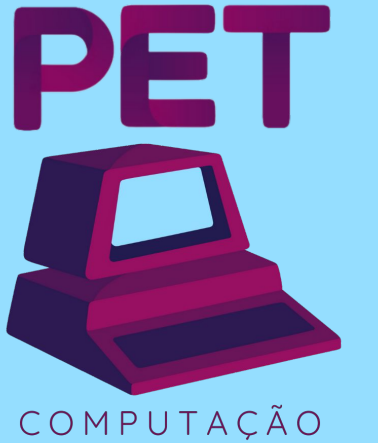
Semana de Calouros 2023

Linux

Aula I

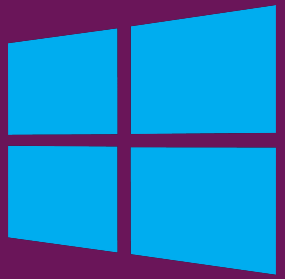
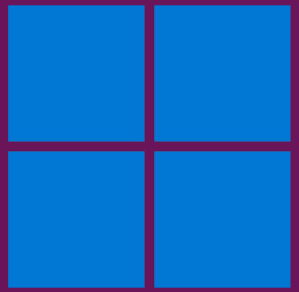
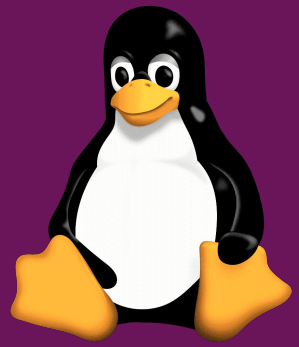


Login: petsemcalX
Senha: PetSemcal#X

2023

1.

Sistemas Operacionais



Árvore de Sistemas Operacionais

Sistema Operacional

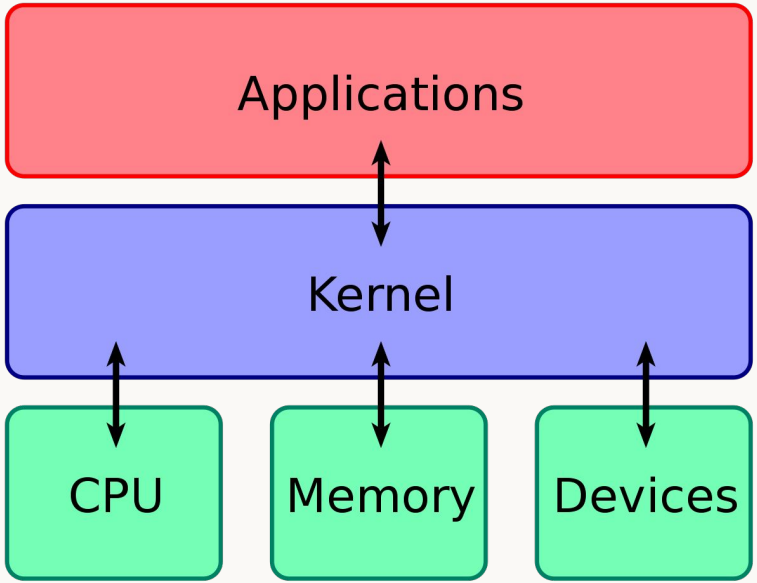
“O sistema operacional é uma camada de software que opera entre o hardware e os programas aplicativos voltados ao usuário final. Trata-se de uma estrutura de software ampla, muitas vezes complexa, que incorpora aspectos de baixo nível (como drivers de dispositivos e gerência de memória física) e de alto nível (como programas utilitários e a própria interface gráfica).” — Carlos Maziero

Sistemas Operacionais - Maziero

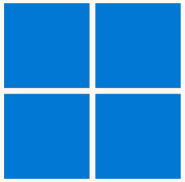
Abstração e Gerência

Kernel

- O SO é dividido em diversas partes
- O kernel (núcleo) é a parte mais baixo nível do sistema operacional
- Gerencia recursos do sistema, hardware etc.
- Outra possível parte de um **SO** é o conjunto de programas utilitários disponibilizados, como compiladores e editor de texto



Windows



- Desenvolvido pela Microsoft a partir de 1985
- Escrito em C/C++ e C#
- Originado do MS-DOS, feito pela Microsoft e baseado no IBM PC DOS
- Baixa segurança (muitos malware)
- Falta de privacidade (telemetria e coleta de dados)

- Código fechado
- Programas pagos
- Possibilidade de instalação em computadores x86-64
- É preciso apenas comprar licença 🤫

macOS

- Desenvolvido pela Apple a partir de 2001
- Escrito em Objective-C (Kernel em C e Assembly)
- Originado do BSD (Berkeley Software Distribution)
- Falta de privacidade

- Código fechado
- Programas pagos
- Exclusivo para MacBook e outros PCs da Apple
- Hackintosh 🕵️


Microsoft :
Wait, I can explain
Google :
You spy through the whole OS?
I only spy the web.
Apple :
Your users know that
you are spying?
Linux:
You guys are spying?

Seria possível existir um sistema operacional *livre*?

2.

História do Linux

Unix Uniplexed Information and Computing Service

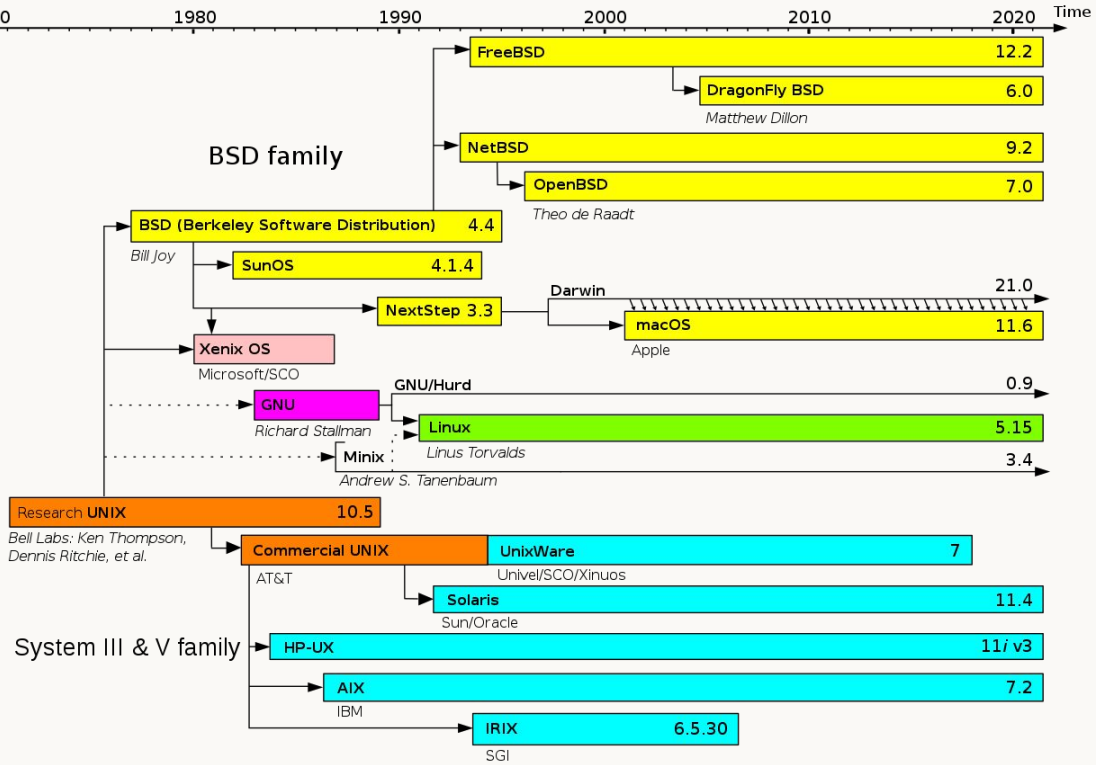
UNIX®
An Open Group Standard

- Sistema operacional desenvolvido pela Bell Labs da AT&T a partir de 1969
- Criado por Ken Thompson e Dennis Ritchie (desenvolvedores das linguagens B e C)
- Escrito em Assembly e C
- Código fechado
- 💡 Pesquise sobre a *Unix philosophy*

- Oferece comandos simples que podem ser combinados com **pipes**
- Shell scripting
- Deu origem a variantes:
  - BSD → FreeBSD e macOS
  - Solaris
  - IBM AIX

Unix Uniplexed Information and Computing Service

UNIX®
An Open Group Standard



Unix Uniplexed Information and Computing Service

UNIX®
An Open Group Standard

Contudo, o Unix disponibilizado pela AT&T não era *livre*.

Ken Thompson e Dennis Ritchie

GNU GNU is **N**ot **U**nix

- Sistema operacional baseado no comportamento do Unix (**Unix-like**)
- Criado por Richard Stallman e comunidade
- Centrado no conceito de *Free Software*
- A comunidade do GNU se concentrou em desenvolver principalmente programas utilitários disponibilizados pelo sistema operacional
- Ex: **gcc** *(GNU Compiler Collection)*

- O kernel do GNU, **Hurd**, não atraiu atenção o suficiente para seu desenvolvimento, ficando incompleto
- 💡 Pesquise sobre a *GNU General Public License* (GPL)

GNU — Free Software

FREE SOFTWARE FOUNDATION

- “meios técnicos para um fim **social**”
- *Free* não está para gratuito, mas para **livre**
- Liberdades fundamentais:
  - Poder utilizar um software para **qualquer** propósito
  - Poder estudar o código e **modificá-lo**
  - Poder **redistribuir** cópias, cobrando ou não
  - Poder **distribuir** cópias de **sua** versão modificada
- Ou seja, acesso ao **código-fonte** (*open-source*)
- 💡 Pesquise sobre o **movimento software livre** e o **Manifesto GNU**

Richard Stallman

E o kernel?

Minix Minimal Unix

MINIX

- Sistema operacional ***Unix-like*** desenvolvido por Andrew Tanenbaum a partir de 1987, em C, para fins educacionais
- Apresentava uma arquitetura de microkernel, ou seja, seu núcleo possui o mínimo de código para funcionar
- Tanenbaum escreveu o livro *Operating Systems: Design and Implementation* com um resumo do código do kernel e outros componentes

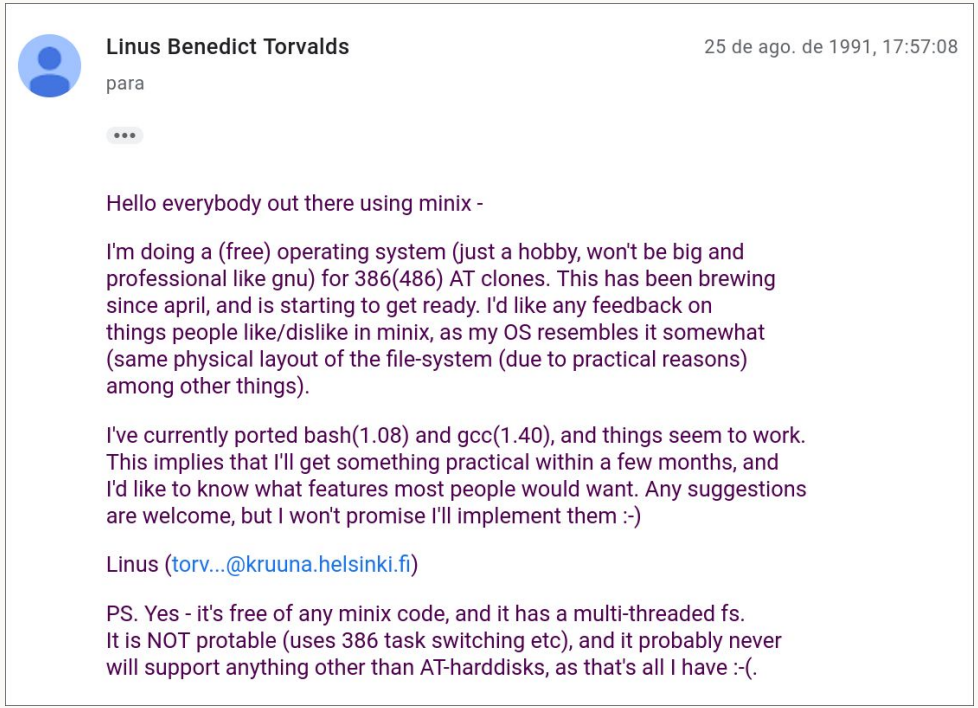
Andrew Tanenbaum

Linus Torvalds

- Cientista da computação finlandês formado pela Universidade de Helsinki
- Na universidade, fez um curso sobre Unix e leu o livro de Tanenbaum sobre o Minix
- Mesmo com o código do Minix disponível, sua modificação e redistribuição não era permitida
- Interessado, Linus iniciou um projeto...

Linus Torvalds

What would you like to see most in minix?



Linus Benedict Torvalds 25 de ago. de 1991, 17:57:08
para

•••

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torv...@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs.
It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

"Olá a todos que estão usando minix -

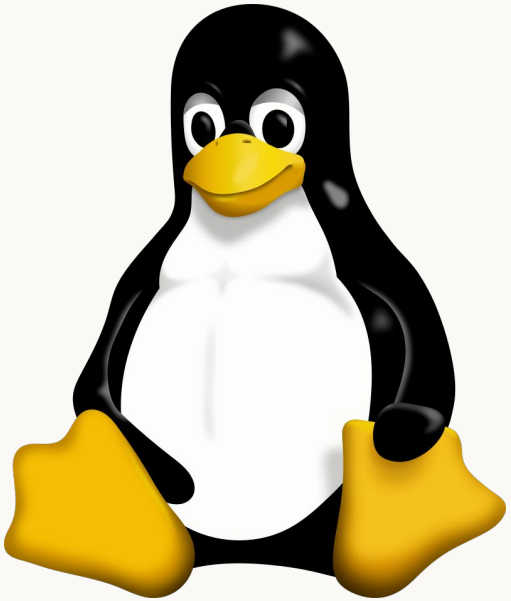
Estou fazendo um sistema operacional livre (apenas um hobby, não será grande e profissional como GNU) para clones AT-386. Está sendo desenvolvido desde abril, e está começando a ficar pronto. Gostaria de opiniões sobre o que as pessoas gostam e desgostam no minix, já que meu SO se assemelha a ele (mesmo layout físico do sistema de arquivos dentre outras coisas).

[...]"

https://groups.google.com/forum/#!msg/comp.os.minix/dlNtH7RRrGA/SwRavCzVE7gJ

Linux

- Kernel de sistema operacional livre desenvolvido por Linus Torvalds a partir de 1991
- Escrito em C, em 2021 chegou a 27.8 milhões de linhas de código
- Em 5 de outubro de 1991 sua primeira versão oficial foi lançada, período em que o Linux já conseguia rodar alguns softwares do GNU, como *bash* e *gcc*
- Inicialmente seu código era escrito apenas por Torvalds, após isso, se tornou um software com contribuições de diversas empresas e milhares de pessoas
- Em fevereiro de 1992, Linux foi licenciado sob a GPL, tornando-se um software livre



Tux (mascote do Linux)

Recapitulando...

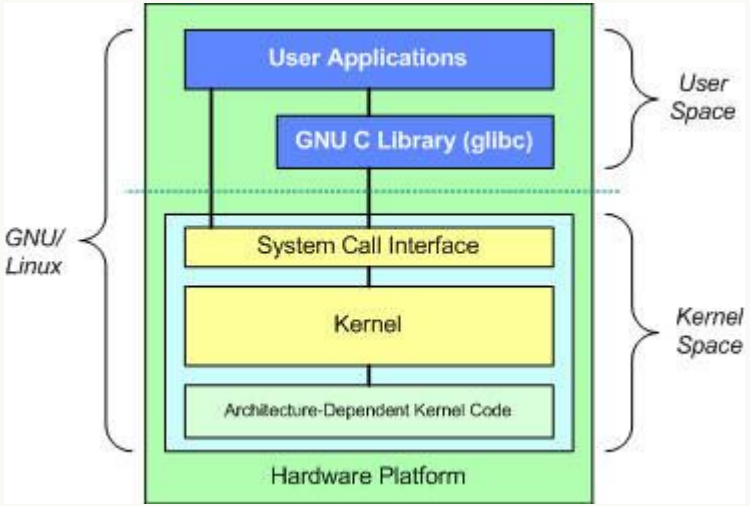
GNU, um sistema operacional livre com diversos programas

Linux, um kernel livre e poderoso

Nasce GNU/Linux

GNU/Linux ou apenas **Linux**

- Sistema operacional *Unix-like* livre e *open-source* que utiliza o kernel Linux e programas (pacotes) do GNU
- GNU não conta apenas com programas para usuário final, como o GIMP, mas também com inúmeros programas para desenvolvedores
- Contudo, não há "**o** sistema operacional Linux", isto é, Linux é uma **família** de diversos sistemas operacionais



… e que sistemas operacionais são esses?!

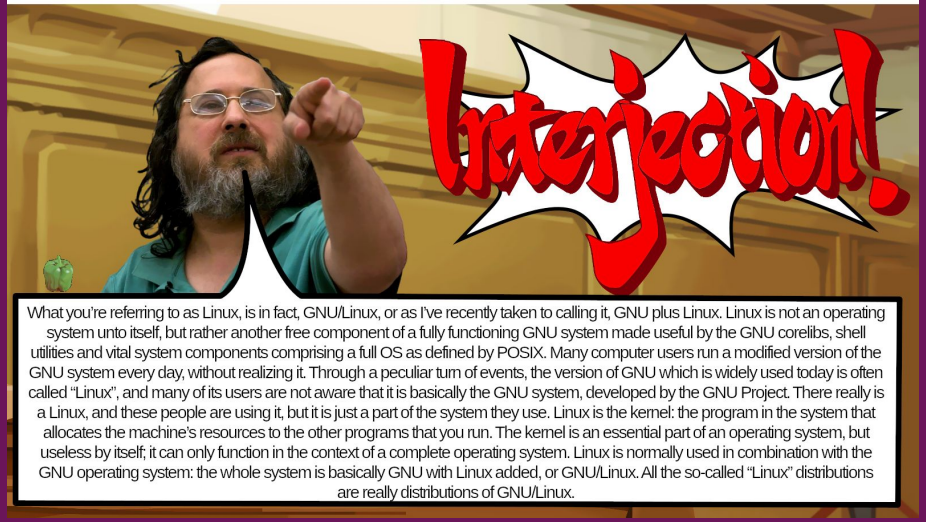
Vantagens do GNU/Linux

- Gratuito
- Mais seguro
- Código-livre, que permite a comunidade a checar o código
- Mais liberdade para modificações
- Melhor ambiente para programação

Estatísticas

- Os 500 melhores supercomputadores do mundo rodam Linux
- 85% de todos os smartphones do mundo usam Linux
- Dos 25 maiores sites do mundo, 23 são hospedados em servidores que rodam Linux, como Facebook, Youtube e Intel
- 75% dos PCs rodam Windows, 15% macOS e 2,14% Linux
- Em PCs de profissionais de computação, Linux chega a mais de 40%

When someone calls it "Linux"



"Linux" vs. "GNU/Linux"

3.

Distribuições

Distribuição ou **distro**

- Uma distribuição Linux é um sistema operacional que utiliza o kernel Linux e os pacotes do GNU
- Também, adiciona outros programas funcionais para um sistema operacional
- Ex: sistema gerenciador de pacotes e interface gráfica
- (Um gerenciador de pacotes é um programa que automatiza o processo de instalação, atualização e remoção de outros programas)

- As distribuições Linux se diferem a partir de softwares que utilizam para compor o sistema
- Desta forma, surgiram inúmeras distribuições, formando diversas famílias a partir de modificações feitas pela comunidade
- Uma distribuição pode ter versões definidas ou ser atualizada constantemente (*rolling-release*)

Árvore de distribuições GNU/Linux

Principais famílias

Debian

- Sua primeira versão foi lançada em 1993
- Por padrão, utiliza o gerenciador de pacotes **apt**
- Utilizado bastante em servidores, devido a sua estabilidade
- Seus principais filhos, como o Ubuntu, são voltados ao usuário comum, visto sua interface amigável
- Assim, é família mais popular para o público geral

debian

Fedora

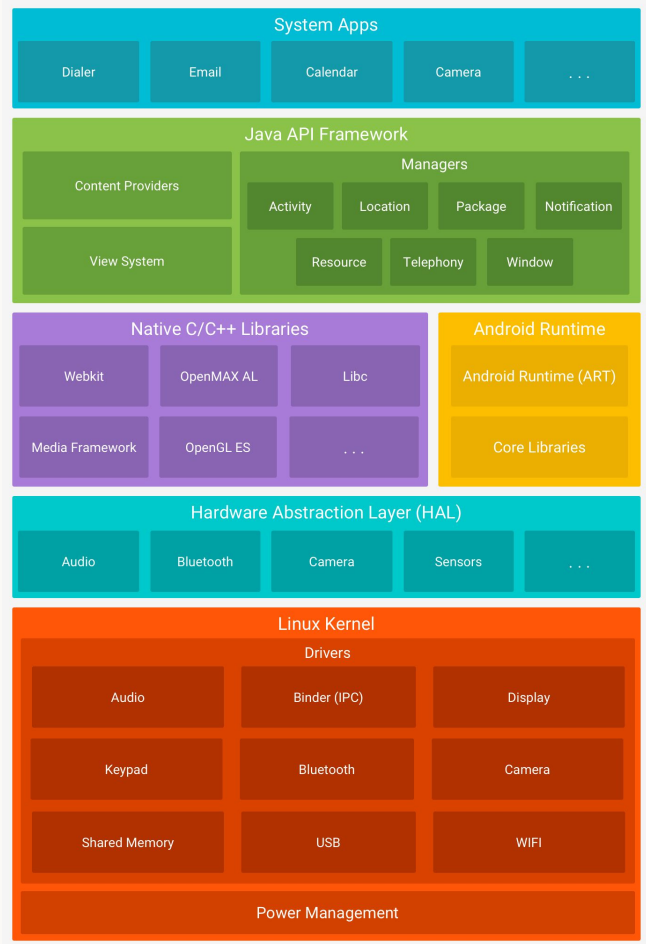
- Sua primeira versão foi lançada em 2003
- Criada pela empresa Red Hat, após a descontinuação do **Red Hat Linux**
- Sua irmã, a **Red Hat Enterprise Linux**, é voltada para a utilização em ambientes corporativos
- Por padrão, utiliza o gerenciador de pacotes **rpm**

Arch

- Lançamento inicial em 2002
- Não possui versões definidas, é uma distribuição **rolling-release**, ou seja, é atualizado constantemente para possuir as versões mais recentes dos softwares instalados
- Utiliza o gerenciador de pacotes **pacman**
- Por padrão, vem com o mínimo de software instalado, e cabe ao usuário montar seu próprio ambiente
- Possui a filha **manjaro** como um sistema amigável, que possui alguns softwares instalados por padrão

Curiosidade: Android

- Android é um sistema operacional mobile *open-source* desenvolvido principalmente pela Google
- Apesar de não ser considerada uma distribuição GNU/Linux, utiliza uma versão modificada do kernel Linux em seu interior



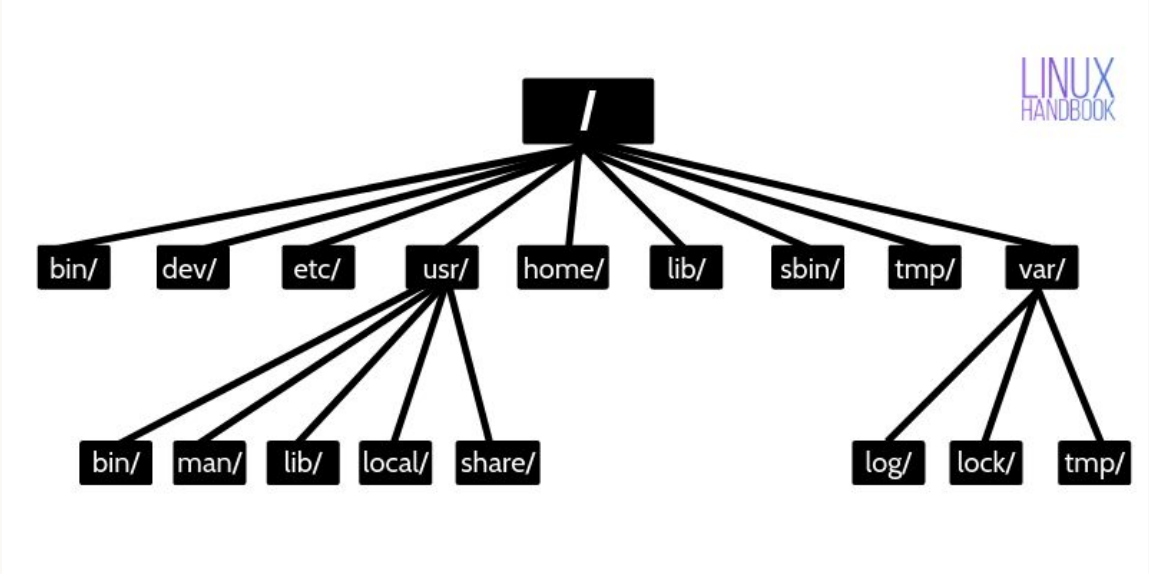
4.

Sistema de Arquivos

Sistema de arquivos

- Gerenciamento do espaço de armazenamento do disco
- Organização da memória em arquivos contidos em diretórios (pastas)
- Cada diretório pode conter arquivos e outros diretórios
- Diretório e arquivo possuem nome (*case-sensitive*)
- Links simbólicos (atalho)

Sistema de arquivos



Sistema de arquivos não precisa decorar!

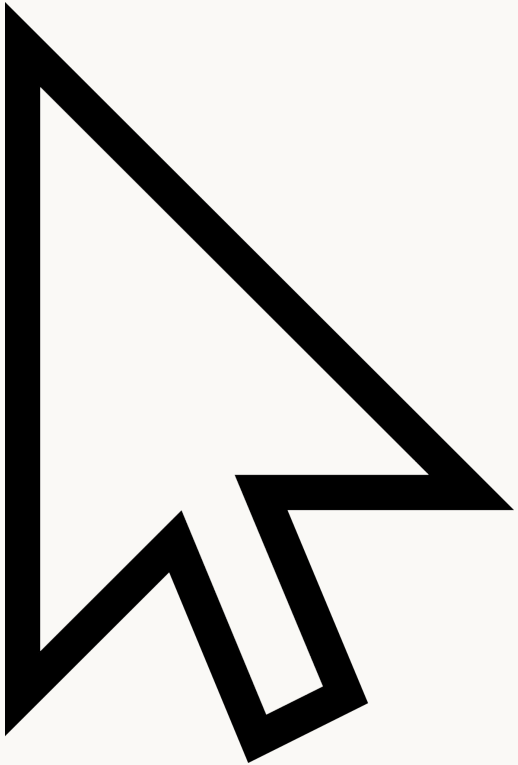
- / → diretório raiz (*root*), todos diretórios e arquivos do SO estão contidos nele
- /bin → arquivos binários executáveis, como comandos da shell
- /dev → arquivos virtual especiais, como aqueles que representam dispositivos físicos
- /etc → arquivos de configuração do sistema
- /usr → arquivos executáveis dos **usuários**, bibliotecas, códigos-fonte e documentação
- /home → diretórios pessoais de cada usuário
- /lib → as bibliotecas necessárias para os executáveis de /bin e /sbin
- /sbin → binários do sistemas, só podem ser executados pelo **root**
- /tmp → arquivos temporários, conteúdo excluído ao reiniciar o computador
- /var → arquivos de dados variáveis, programas armazenam dados de tempo de execução
- /boot → arquivos do kernel e de boot
- /proc → arquivos sobre processos
- /opt → arquivos de aplicações opcionais (não disponíveis pela distribuição)
- /root → diretório pessoal do **usuário** root
- /media → diretórios montados para mídias removíveis (USB, DVD, etc.)

5.

Interface Gráfica

GUI **G**raphical **U**ser **I**nterface

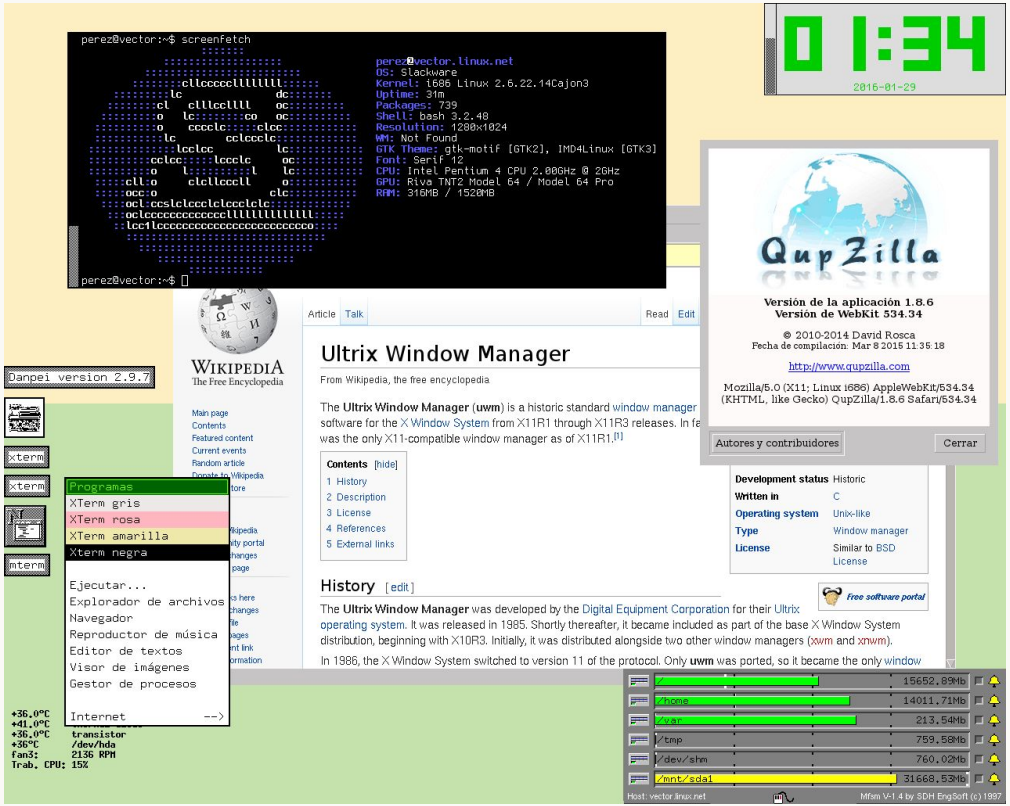
- Interface visual para usuários interagirem com os dispositivos através de componentes gráficos
- Alternativa ao **CLI** (*command-line interface*)
- Menor curva de aprendizagem em relação a comandos por texto
- Assim, surgem os programas com interface gráfica e suas janelas (*windows*)



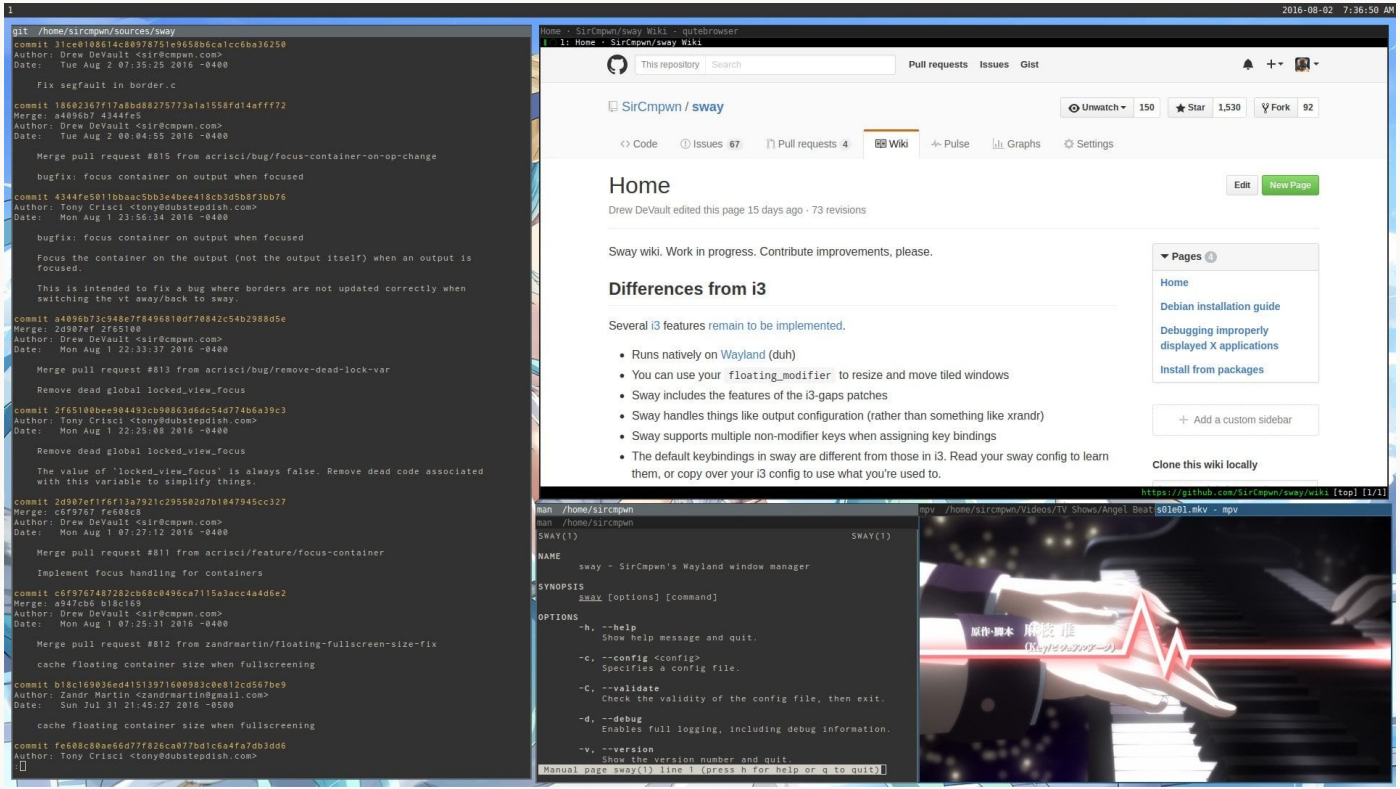
Window Manager

- Software do sistema que controla a localização, comportamento e aparência das janelas de programas
- Determina tamanho e bordas das janelas, dentre outras características
- **Pode** conter menus que iniciam outros programas, assim como barras de status com horário, data e outras informações
- Faz parte de *desktop environments*, mas pode ser utilizado sozinho
- Sendo utilizado sozinho, faz grande uso de atalhos de teclado para gerenciamento das janelas
- Há três tipos de WMs: *stacking*, *tiling* e dinâmica (combina as duas primeiras)
- 💡 Pesquise sobre Xorg e Wayland

Stacking


perez@vector:~$ screenfetch
perez@vector.linux.net
OS: Slackware
Kernel: 1686 Linux 2.6.22.14Cajon3
Uptime: 34m
Packages: 739
Shell: bash 3.2.48
Resolution: 1280x1024
WM: Not Found
GTK Theme: gtk-motif [GTK2], IMD4Linux [GTK3]
Font: Serif 12
CPU: Intel Pentium 4 CPU 2.80GHz @ 2GHz
GPU: Riva TNT2 Model 64 / Model 64 Pro
RAM: 316MB / 1520MB
perez@vector:~$
01:34
2016-01-29
QupZilla
Versión de la aplicación 1.8.6
Versión de WebKit 534.34
© 2010-2014 David Rosca
Fecha de compilación: Mar 8 2016 11:35:18
http://www.qupzilla.com
Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.34 (KHTML, like Gecko) QupZilla/1.8.6 Safari/534.34
Autores y contribuidores
Cerrar
Danpei version 2.9.7
xterm
xterm
mterm
Programas
XTerm gris
XTerm rosa
XTerm amarilla
Xterm negra
Ejecutar...
Explorador de archivos
Navegador
Reproductor de música
Editor de textos
Visor de imágenes
Gestor de procesos
Internet
WIKIPEDIA
The Free Encyclopedia
Article
Talk
Read
Edit
Ultrix Window Manager
From Wikipedia, the free encyclopedia
The Ultrix Window Manager (uwm) is a historic standard window manager software for the X Window System from X11R1 through X11R3 releases. In fa
was the only X11-compatible window manager as of X11R1.[1]
Contents [hide]
1 History
2 Description
3 License
4 References
5 External links
History [edit]
The Ultrix Window Manager was developed by the Digital Equipment Corporation for their Ultrix operating system. It was released in 1985. Shortly thereafter, it became included as part of the base X Window System distribution, beginning with X10R3. Initially, it was distributed alongside two other window managers (xwm and xnwm).
In 1986, the X Window System switched to version 11 of the protocol. Only uwm was ported, so it became the only window
Development status Historic
Written in C
Operating system Unix-like
Type Window manager
License Similar to BSD License
Free software portal
+36.0°C
+41.0°C
+36.0°C transistor
+36°C /dev/hda
fan3: 2136 RPM
Trab. CPU: 15%
/ 15652.89Mb
/home 14011.71Mb
/var 213.54Mb
/tmp 759.58Mb
/dev/shm 760.02Mb
/mnt/sda1 31668.53Mb
Host: vector.linux.net
Mfism V-1.4 by SDH EngSoft (c) 1997

Tiling


1
2016-08-02 7:36:50 AM
git /home/sircmpwn/sources/sway
commit 31ce0108614c80978751e9658b6ca1cc6ba36250
Author: Drew DeVault <sir@cmpwn.com>
Date: Tue Aug 2 07:35:25 2016 -0400
Fix segfault in border.c
commit 18602367f17a8bd88275773a1a1558fd14afff72
Merge: a4096b7 4344fe5
Author: Drew DeVault <sir@cmpwn.com>
Date: Tue Aug 2 00:04:55 2016 -0400
Merge pull request #815 from acrisci/bug/focus-container-on-op-change
bugfix: focus container on output when focused
commit 4344fe5011bbaac5bb3e4bee418cb3d5b8f3bb76
Author: Tony Crisci <tony@dubstepdish.com>
Date: Mon Aug 1 23:56:34 2016 -0400
bugfix: focus container on output when focused
Focus the container on the output (not the output itself) when an output is focused.
This is intended to fix a bug where borders are not updated correctly when switching the vt away/back to sway.
commit a4096b73c948e7f8496810df70842c54b2988d5e
Merge: 2d907ef 2f65100
Author: Drew DeVault <sir@cmpwn.com>
Date: Mon Aug 1 22:33:37 2016 -0400
Merge pull request #813 from acrisci/bug/remove-dead-lock-var
Remove dead global locked_view_focus
commit 2f65100bee904493cb90863d6dc54d774b6a39c3
Author: Tony Crisci <tony@dubstepdish.com>
Date: Mon Aug 1 22:25:08 2016 -0400
Remove dead global locked_view_focus
The value of 'locked_view_focus' is always false. Remove dead code associated with this variable to simplify things.
commit 2d907ef1f6f13a7921c295502d7b1047945cc327
Merge: c6f9767 fe608c8
Author: Drew DeVault <sir@cmpwn.com>
Date: Mon Aug 1 07:27:12 2016 -0400
Merge pull request #811 from acrisci/feature/focus-container
Implement focus handling for containers
commit c6f9767487282cb68c0496ca7115a3acc4a4d6e2
Merge: a947cb6 b18c169
Author: Drew DeVault <sir@cmpwn.com>
Date: Mon Aug 1 07:25:31 2016 -0400
Merge pull request #812 from zandrmartin/floating-fullscreen-size-fix
cache floating container size when fullscreening
commit b18c169036ed41513971600983c0e812cd567be9
Author: Zandr Martin <zandrmartin@gmail.com>
Date: Sun Jul 31 21:45:27 2016 -0500
cache floating container size when fullscreening
commit fe608c80ae66d77f826ca077bd1c6a4fa7db3dd6
Author: Tony Crisci <tony@dubstepdish.com>
Home · SirCmpwn/sway Wiki - qutebrowser
1: Home · SirCmpwn/sway Wiki
This repository
Search
Pull requests
Issues
Gist
SirCmpwn / sway
Unwatch 150
Star 1,530
Fork 92
Code
Issues 67
Pull requests 4
Wiki
Pulse
Graphs
Settings
Home
Edit
New Page
Drew DeVault edited this page 15 days ago · 73 revisions
Sway wiki. Work in progress. Contribute improvements, please.
Differences from i3
Several i3 features remain to be implemented.
Runs natively on Wayland (duh)
You can use your floating_modifier to resize and move tiled windows
Sway includes the features of the i3-gaps patches
Sway handles things like output configuration (rather than something like xrandr)
Sway supports multiple non-modifier keys when assigning key bindings
The default keybindings in sway are different from those in i3. Read your sway config to learn them, or copy over your i3 config to use what you're used to.
Pages 4
Home
Debian installation guide
Debugging improperly displayed X applications
Install from packages
Add a custom sidebar
Clone this wiki locally
https://github.com/SirCmpwn/sway/wiki [top] [1/1]
man /home/sircmpwn
man /home/sircmpwn
SWAY(1) SWAY(1)
NAME
sway - SirCmpwn's Wayland window manager
SYNOPSIS
sway [options] [command]
OPTIONS
-h, --help
Show help message and quit.
-c, --config <config>
Specifies a config file.
-C, --validate
Check the validity of the config file, then exit.
-d, --debug
Enables full logging, including debug information.
-v, --version
Show the version number and quit.
Manual page sway(1) line 1 (press h for help or q to quit)
mpv /home/sircmpwn/Videos/TV Shows/Angel Beats/s01e01.mkv - mpv

Desktop Environment

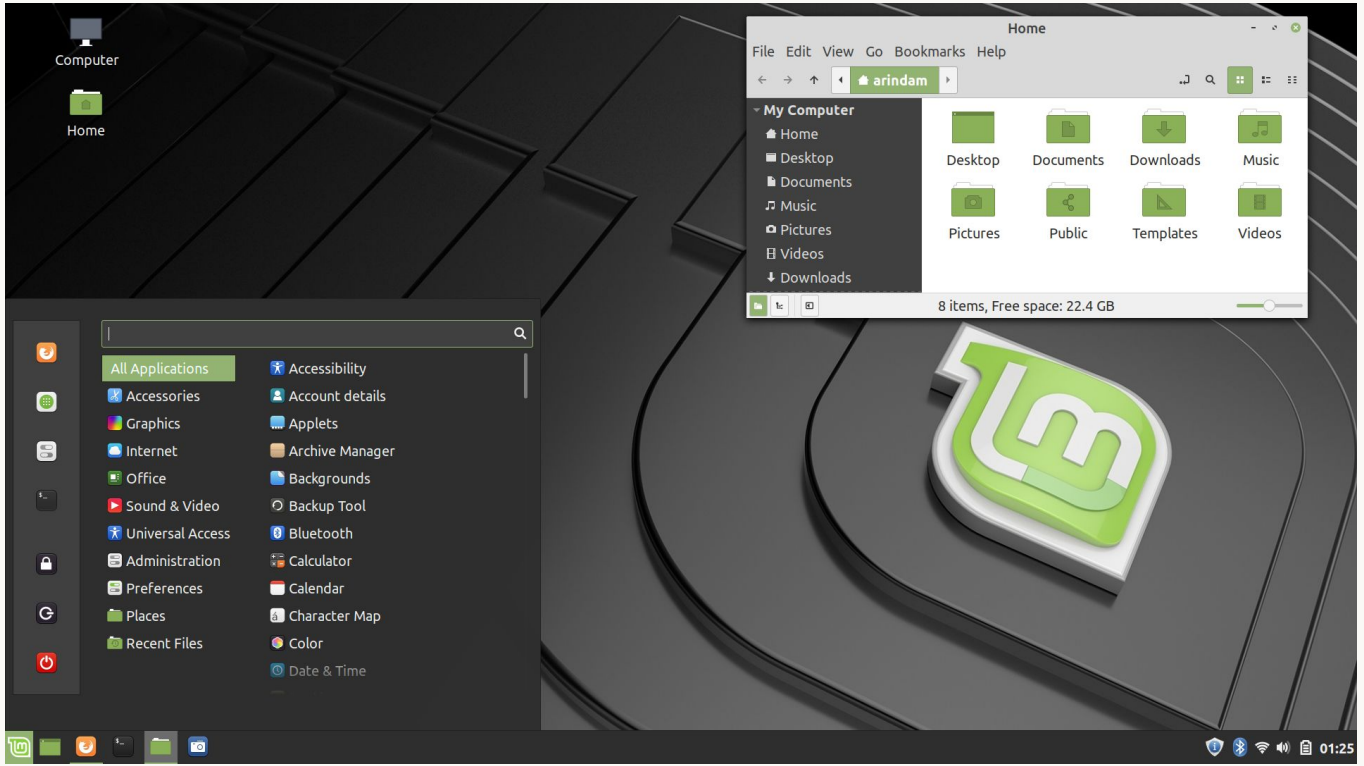
- Contém um *window manager*, adicionando (muitas) outras funcionalidades
- Geralmente, adiciona:
  - Ícones de atalhos para iniciar programas
  - Barras de ferramentas
  - Widgets na área de trabalho
  - Conjunto de programas utilitários (ex: office e editor de texto com GUI)

Exemplos de *Desktop Environments*

GNOME

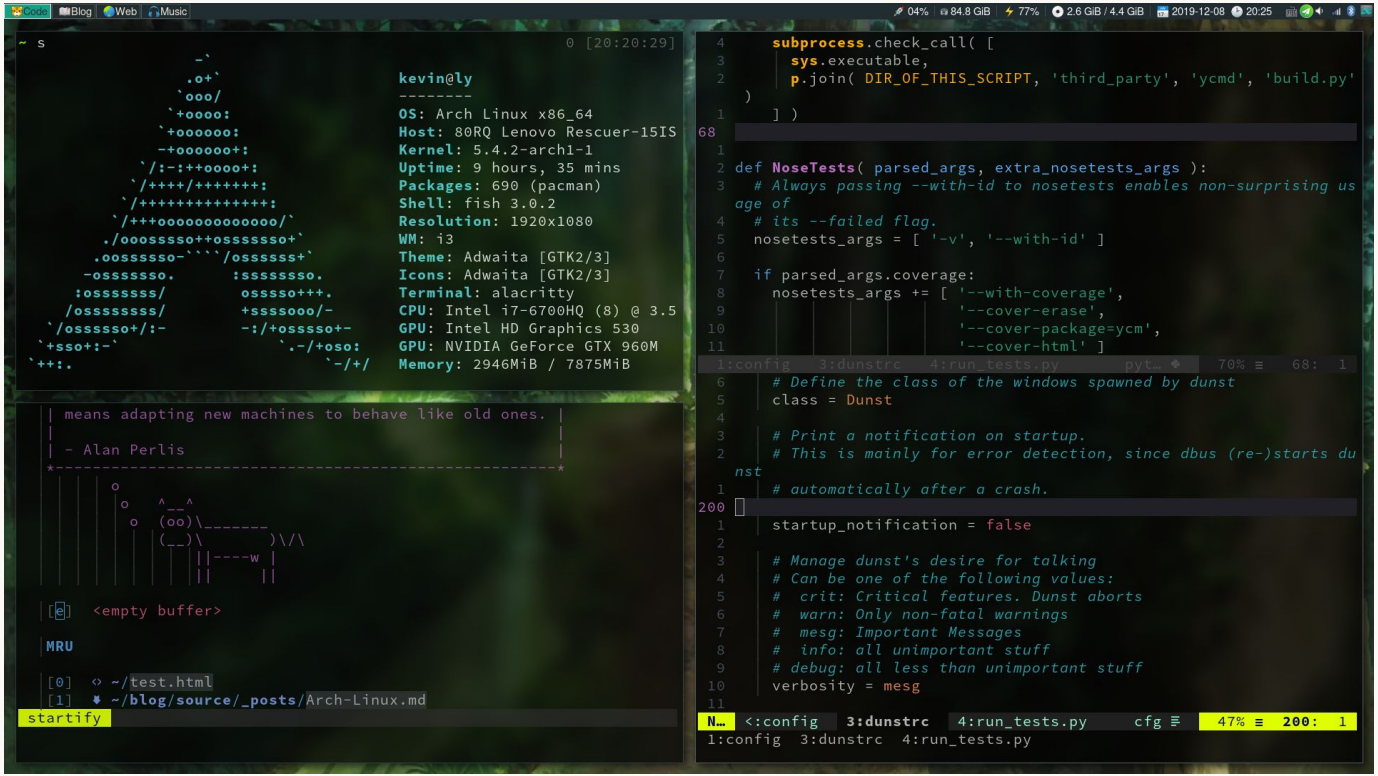

Activities
Wed 13:40
Type to search
Wired Connected
Location Disabled
Estimating...
NoobsLab.com
Activity Log ...
AisleRiot Soli...
Amazon
aTunes
Ayatana Indi...
Boostnote
Browser
Calculator
Calendar
Cerebro
Checkbox
Cheese
Clipboard Au...
ClipGrab
Contacts
DiskStat Indi...
Files
Files Indicator
Firefox Web ...
flowblade
Galculator
Gnome-Pie
Google Earth
ImageMagic...
Frequent
All

Cinnamon


Computer
Home
Home
File Edit View Go Bookmarks Help
arindam
My Computer
Home
Desktop
Documents
Music
Pictures
Videos
Downloads
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
8 items, Free space: 22.4 GB
All Applications
Accessories
Graphics
Internet
Office
Sound & Video
Universal Access
Administration
Preferences
Places
Recent Files
Accessibility
Account details
Applets
Archive Manager
Backgrounds
Backup Tool
Bluetooth
Calculator
Calendar
Character Map
Color
Date & Time
01:25

Exemplos de *Window Managers*

i3


Code
Blog
Web
Music
04%
84.8 GiB
77%
2.6 GiB / 4.4 GiB
2019-12-08
20:25
~ s
0 [20:20:29]
kevin@ly
OS: Arch Linux x86_64
Host: 80RQ Lenovo Rescuer-15IS
Kernel: 5.4.2-arch1-1
Uptime: 9 hours, 35 mins
Packages: 690 (pacman)
Shell: fish 3.0.2
Resolution: 1920x1080
WM: i3
Theme: Adwaita [GTK2/3]
Icons: Adwaita [GTK2/3]
Terminal: alacritty
CPU: Intel i7-6700HQ (8) @ 3.5
GPU: Intel HD Graphics 530
GPU: NVIDIA GeForce GTX 960M
Memory: 2946MiB / 7875MiB
means adapting new machines to behave like old ones.
- Alan Perlis
[e] <empty buffer>
MRU
[0] ~/test.html
[1] ~/blog/source/_posts/Arch-Linux.md
startify
subprocess.check_call([
sys.executable,
p.join(DIR_OF_THIS_SCRIPT, 'third_party', 'ycmd', 'build.py'
])
def NoseTests(parsed_args, extra_nosetests_args):
# Always passing --with-id to nosetests enables non-surprising us
age of
# its --failed flag.
nosetests_args = ['-v', '--with-id']
if parsed_args.coverage:
nosetests_args += ['--with-coverage',
'--cover-erase',
'--cover-package=ycm',
'--cover-html']
1:config 3:dunstrc 4:run_tests.py
70% ≡ 68: 1
# Define the class of the windows spawned by dunst
class = Dunst
# Print a notification on startup.
# This is mainly for error detection, since dbus (re-)starts du
nst
# automatically after a crash.
startup_notification = false
# Manage dunst's desire for talking
# Can be one of the following values:
# crit: Critical features. Dunst aborts
# warn: Only non-fatal warnings
# mesg: Important Messages
# info: all unimportant stuff
# debug: all less than unimportant stuff
verbosity = mesg
N… <:config 3:dunstrc 4:run_tests.py cfg ≡
47% ≡ 200: 1
1:config 3:dunstrc 4:run_tests.py

Hyprland


1 2 3 4
Motley Crew - Post Malone
6.0.10-linuz2-1
Nest Wi-Fi_5G
69°C
18%
100%
50%
07:56 AM

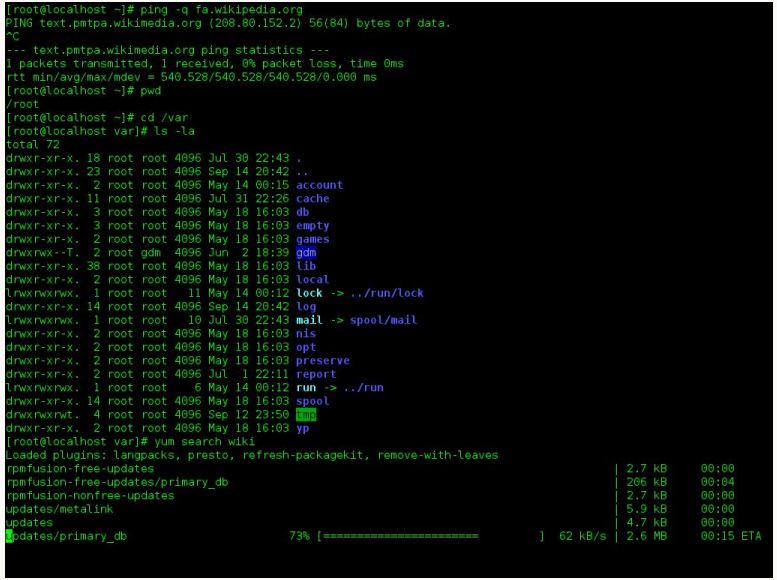

Most linux distro reviews on YouTube are really only desktop environment reviews

6.

Terminal e Shell

CLI Command-Line Interface

- Interface que recebe **comandos** do usuário em forma de linhas de texto
- Através de comandos, o usuário pode alterar o ambiente e utilizar programas
- Maior poder de automatização
- Mais rapidez para executar tarefas
- … mais difícil para usuários leigos?



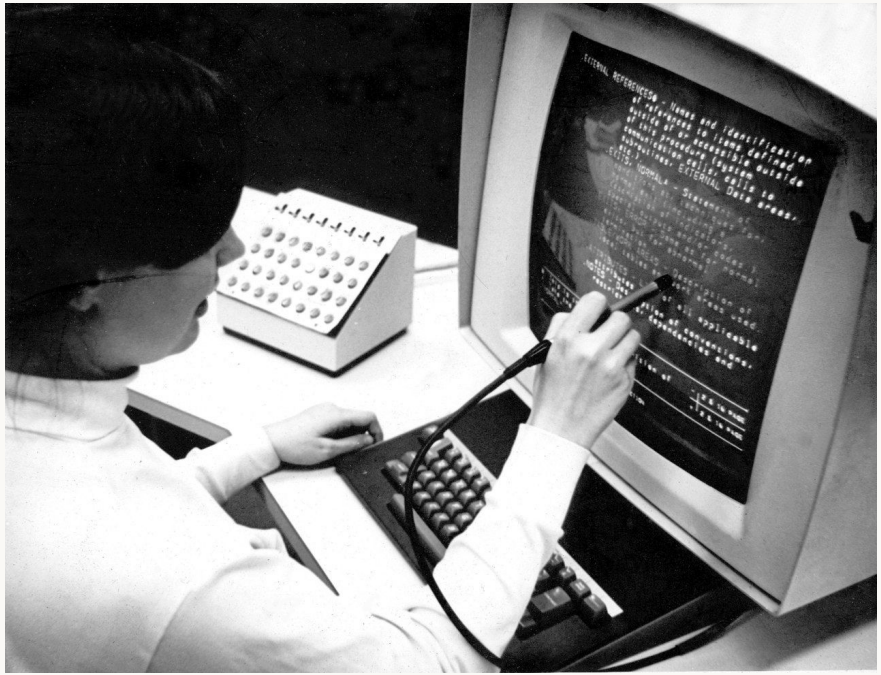
Terminal

- Dispositivo eletrônico ou eletromecânico que permite a entrada e saída de dados
- Os primeiros terminais eram os teletipos (*Teletypes*), máquinas de escrever que enviava e recebia mensagens
- Os TTY eram telégrafos que enviavam mensagens escritas em seu teclado (como máquina de escrever)
- Recebia mensagens e imprimia em papel

Teleprinter / *Teletype* / **TTY** usado na WW2

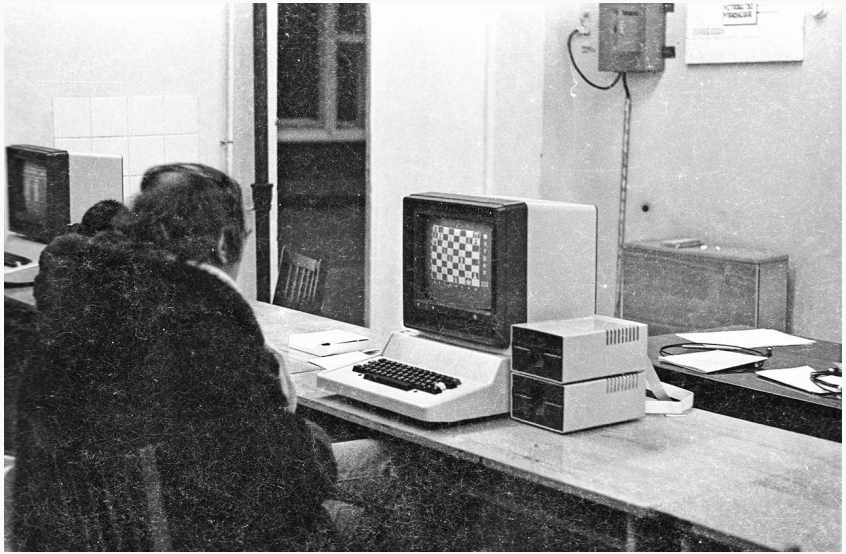
Terminal

- Nos anos 50, a impressão em papel começou a ser substituída por displays de vídeo
- Nos anos 60, terminais inteligentes foram lançados, os quais possuíam processamento próprio a partir de microprocessadores
- Apesar de poder fazer algum processamento de dados, estes terminais eram conectados a um computador externo, como um *mainframe*, que executava os comandos digitados no terminal



Computadores pessoais

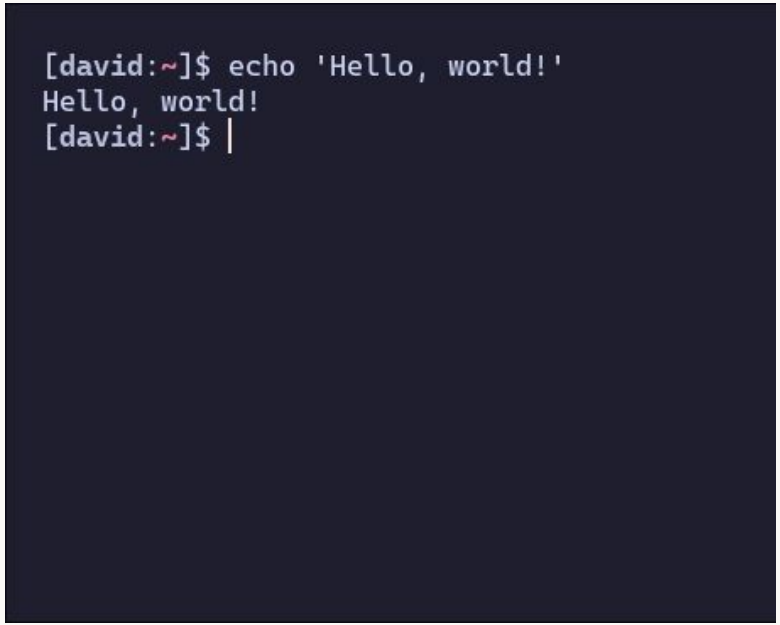
- Diversos terminais eram conectados a cada computador, exigindo grande processamento
- Com o barateamento de CPUs, os terminais foram ficando cada vez mais poderosos
- Com isso, surgem os PCs (*Personal Computers*), criados para uso pessoal



Computador *Pravetz* 82, utilizado na União Soviética

Emuladores de terminal

- Os PCs ofereciam terminais, mas estes não eram como os terminais anteriores
- Visto que um PC não possui um hardware para a geração de caracteres, mas sim placas de vídeo, os terminais são **emulados**
- Emuladores de terminal são programas que emulam terminais de vídeo, dentro de uma arquitetura de *display*



Shell

- Software que interpreta comandos de texto
- Linguagem de comandos interativa
- Linguagem de scripting
- É acessada utilizando um emulador de terminal
- Ex: gnome terminal, terminator, kitty, st

- Permite a combinação de diversos comandos (programas)
- bash: shell da GNU
- zsh: shell que adiciona outras funcionaliades além das presentes no bash

Working directory

- Ao iniciar um terminal, por padrão ele abrirá dentro do diretório pessoal do usuário ativo (ex: /home/pet)
- É possível navegar pelos diretórios do sistema de arquivos a partir de comandos
- Assim, o diretório que o usuário se encontra no terminal em um determinado momento é chamado de *working directory* (diretório de trabalho)

Abra o terminal com o atalho Ctrl+Alt+T

Path ou caminho

- Nos comandos interpretados pelo shell, usamos **caminhos** para representar uma localização do sistema de arquivos
- Um ***path*** é um texto que representa a localização de um arquivo ou diretório
- Os caminhos podem ser absolutos ou relativos
- O caminho absoluto aponta para o mesmo local, independentemente do diretório de trabalho atual
- Assim, para representar uma localização com o caminho absoluto deve-se utilizar o diretório raiz (o barra “/”)
- /home/pet/Pictures/Wallpapers/wallpaper.png
- No caminho relativo, representa-se uma localização **relativa** ao *working directory*
- Ou seja, a localização de um caminho relativo pode mudar, a depender do *working directory*
- Pictures/Wallpapers/wallpaper.png (assumindo que o diretório atual é /home/pet)

Atalhos da shell

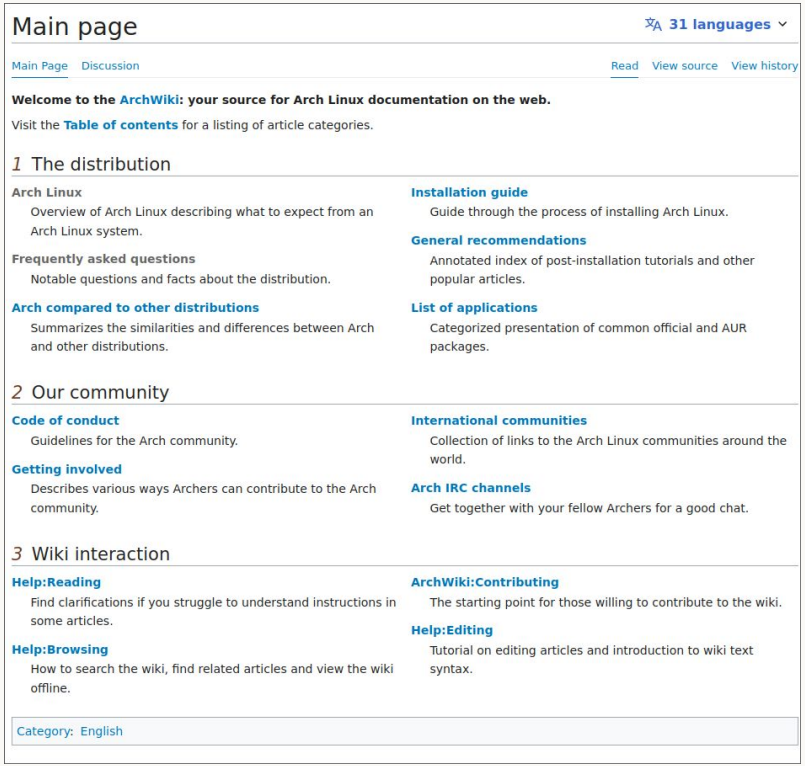
- Como o *working directory* varia bastante ao navegar pelo sistema de arquivos, utiliza-se atalhos (caracteres coringas) em *paths*
- Esses caracteres permitem utilizar o caminho relativo sem precisar alterar o diretório atual
- Caracteres coringas:
  - / → diretório raiz e separação de diretórios
  - ~ → diretório pessoal (home) do usuário atual (ex: /home/pet)
  - . → diretório atual
  - .. → diretório pai do atual
- Assim, ao utilizar os caracteres coringas em um comando, a shell substitui o caractere para o caminho que o representa

7.

Dicas

Arch Linux Wiki wiki.archlinux.org

- Maior wiki sobre assuntos relacionados ao Linux
- Mantida pela comunidade
- Traduzida para várias línguas



Dual boot — Linux e Windows

- Instalação de dois (ou mais) sistemas operacionais em um computador
- Seleção do sistema operacional a ser executado durante o boot
- Geralmente usa softwares como o GRUB
- 💡 Pesquise sobre *boot loaders*

- Há vários tutoriais no youtube para dual boot com Windows e Linux
- Os passos podem variar de acordo com a distribuição (porém, para a família Debian e Fedora é generalizado)
- Diolinux:
  - https://www.youtube.com/watch?v=6D6L9Wml1oY

Exercício (para casa)

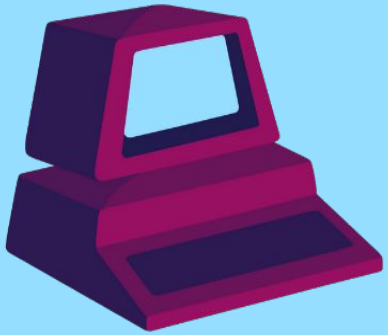
- Revise a aula, com calma, em web.inf.ufpr.br/pet/linux
- Instale alguma distribuição GNU/Linux em seu computador
- Recomendamos alguma distribuição para leigos, como o Ubuntu ou Linux Mint
- Aventure-se com distribuições mais avançadas, como o Arch, quando estiver familiarizado e experiente

Avalie a aula

forms.gle/bJuWakM4pCi6VutR9

Conta como presença!

Obrigado!



PET
COMPUTAÇÃO

pet.inf.ufpr.br

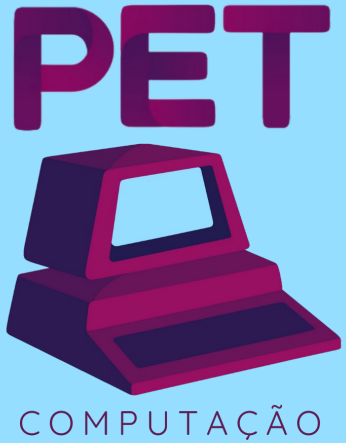
pet@inf.ufpr.br

@petcompufpr

Linux

Aula II

Login: petsemcalX
Senha: PetSemcal#X



2023

Relembrando...

Atalhos da shell

- Como o *working directory* varia bastante ao navegar pelo sistema de arquivos, utiliza-se atalhos (caracteres coringas) em *paths*
- Esses caracteres permitem utilizar o caminho relativo sem precisar alterar o diretório atual
- Caracteres coringas:
  - / → diretório raiz e separação de diretórios
  - ~ → diretório pessoal (home) do usuário atual (ex: /home/pet)
  - . → diretório atual atual
  - .. → diretório pai do atual
- Assim, ao utilizar os caracteres coringas em um comando, a shell substitui o caractere para o caminho que o representa

Atalhos do terminal

- Ctrl+Alt+T → abre o terminal padrão
- Setas (up/down) → navega pelos últimos comandos
- Ctrl+Shift+C/V → copia/cola no terminal
- Ctrl+U → limpa o que está escrito antes do cursor
- Ctrl+L → limpa a tela (como o comando `clear`)
- Tab → *autocomplete* para comandos/argumentos/arquivos quando possível
- Tab Tab → mostra as opções de *autocomplete*
- Ctrl+Z → coloca o processo que está sendo executado em *background*, liberando o *prompt*
- Ctrl+C → mata o processo

1.

Contas no DInf

DInf Departamento de Informática

- Utiliza o **Linux Mint** como sistema operacional nos computadores de seus laboratórios
- Usuário: siglas do nome + ano de ingresso
- Ex: Guilherme Nunes Uzias → gnu23
- Solicitar login e senha de seu usuário na secretaria
- Página web pessoal → `~/public_html`
- Quota de armazenamento
  - /home → 8 GB
  - /nobackup → 8 GB

DInf Departamento de Informática

- Você **não** pode usar **sudo** (se usar, um root receberá um e-mail rs)
- Lembre-se, NÃO FAÇA NADA ILEGAL!
- Você deve alterar a senha do seu usuário, utilize o comando **passwd**
- Não se esqueça da sua senha e não a passe para ninguém!

```
[pet@arch ~]$ passwd
Changing password for pet.
Current password:
New password:
Retype new password:
passwd: password updated successfully
[pet@arch ~]$
```

E-mail do DInf

- Além do e-mail da UFPR (ufpr.br), você terá um e-mail do departamento
- O e-mail será: login + @inf.ufpr.br
  - Ex: gnu23@inf.ufpr.br
- Para acessar o e-mail, entre em webmail.inf.ufpr.br
- O cache do e-mail contará na sua quota
- Você pode utilizar o e-mail por algum cliente de e-mails, como o Thunderbird
  - https://web.inf.ufpr.br/dinf/departamento-de-informatica/infra-estrutura/tecnologica/duvidas-frequentes-sobre-suporte-aos-sistemas

Wi-fi e site pessoal

- No DInf há a disponibilização do Wi-fi C3SL, com 20Gb/s
- Para acessar, siga os passos em:
  https://web.inf.ufpr.br/dinf/departamento-de-informatica/infra-estrutura/tecnologica/duvidas-frequentes-sobre-suporte-aos-sistemas
- Também, você tem acesso a um site pessoal: www.inf.ufpr.br/<login>
  - Ex: www.inf.ufpr.br/gnu23
- Para editá-lo, utilize o diretório public_html
  - Por exemplo, crie um arquivo chamado index.html
  - Acesse-o com www.inf.ufpr.br/<login>/index.html

2.

Comandos básicos

Estrutura de comandos

- Geralmente, um comando segue uma estrutura básica, sendo:

[comando] [opções] [argumentos]

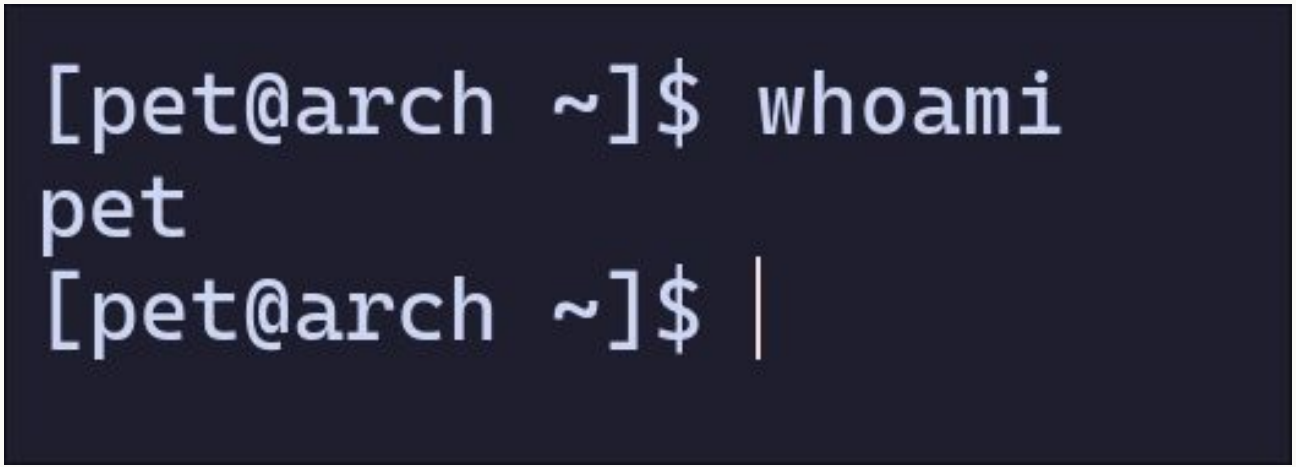
- Exemplo:

cp -r dir1 dir2

ffmpeg -y -i hypr.mp4 -i palette.png -filter_complex -r 10 -s 640x360 hypr.gif

whoami

- Imprime na tela o usuário atual



pwd print working directory

- Imprime na tela o diretório atual (*working directory*)

```
[pet@arch ~]$ pwd
/home/pet
[pet@arch ~]$ cd Pictures/Wallpapers/
[pet@arch Wallpapers]$ pwd
/home/pet/Pictures/Wallpapers
[pet@arch Wallpapers]$
```

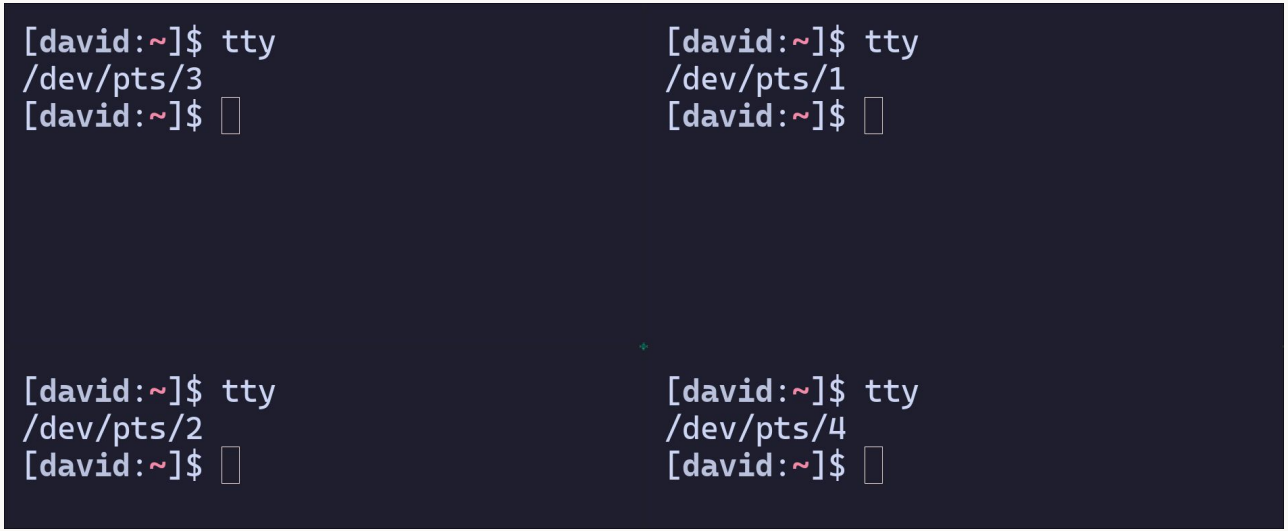
hostname

- Imprime na tela o nome do sistema
- Encontra-se em `/etc/hostname` (“no Linux tudo é um arquivo”)

```
[pet@arch ~]$ hostname
arch
[pet@arch ~]$ cat /etc/hostname
arch
[pet@arch ~]$
```

tty

- Imprime na tela o arquivo do terminal em questão (identificação)



who

- Imprime na tela os usuários conectados ao sistema

```
[david:~]$ who
david    tty1         2023-03-19 12:00
[david:~]$
```

```
dlpg21@macalan:~$ who
sastempliuk pts/0        2023-03-18 15:38 (177.19.7.201)
vwnascimento pts/1        2023-03-19 17:41 (200.24.69.208)
dlpg21   pts/2        2023-03-20 01:03 (2001:1284:f022:b83f
sdominico pts/3        2023-03-19 23:21 (189.123.222.252)
hhyamamura pts/4        2023-03-18 22:33 (tmux(1421223).%0)
dlpg21@macalan:~$
```

Dicas

Histórico

- Utilize as teclas ***up*** e ***down*** para navegar pelo histórico de comandos
- Evita digitar novamente os últimos comandos caso erre

3.

Buscando ajuda

help

- A maioria dos comandos disponibilizam a opção `--help`
- Imprime uma breve documentação sobre o comando
- Também há outras possíveis opções, como `--version`

```
[pet@arch ~]$ pwd --help
pwd: pwd [-LP]
    Print the name of the current working directory.

    Options:
      -L        print the value of $PWD if it names the current working
                directory
      -P        print the physical directory, without any symbolic links

    By default, `pwd' behaves as if `-L' were specified.

    Exit Status:
    Returns 0 unless an invalid option is given or the current directory
    cannot be read.
[pet@arch ~]$
```

man manual

- O comando **man** é usado para ler a documentação completa de algum comando
- Não necessita conexão à internet
- `$ man [comando]`
- Para sair da interface do man, digite **q**
- Teste o comando com:
  - `$ man man`
  - `$ man ls`
  - `$ man cd`

```
MAN(1)                          Manual pager utils                          MAN(1)

NAME
       man - an interface to the system reference manuals

SYNOPSIS
       man [man options] [[section] page ...] ...
       man -k [apropos options] regexp ...
       man -K [man options] [section] term ...
       man -f [whatis options] page ...
       man -l [man options] file ...
       man -w|-W [man options] page ...

DESCRIPTION
       man  is  the system's manual pager.  Each page argument given to man is normally
       the name of a program, utility or function.  The  manual  page  associated  with
       each  of  these  arguments is then found and displayed.  A section, if provided,
       will direct man to look only in that section of the manual.  The default  action
       is to search in all of the available sections following a pre-defined order (see
       DEFAULTS), and to show only the first page found, even if page exists in several
       sections.

       The table below shows the section numbers of the manual followed by the types of
       pages they contain.

       1   Executable programs or shell commands
       2   System calls (functions provided by the kernel)
       3   Library calls (functions within program libraries)
       4   Special files (usually found in /dev)
       5   File formats and conventions, e.g. /etc/passwd
       6   Games
       7   Miscellaneous (including  macro  packages  and  conventions),  e.g.  man(7),
           groff(7), man-pages(7)
       8   System administration commands (usually only for root)
       9   Kernel routines [Non standard]

       A manual page consists of several sections.

       Conventional  section  names include NAME, SYNOPSIS, CONFIGURATION, DESCRIPTION,
       OPTIONS, EXIT STATUS, RETURN VALUE, ERRORS, ENVIRONMENT, FILES,  VERSIONS,  CON-
       FORMING TO, NOTES, BUGS, EXAMPLE, AUTHORS, and SEE ALSO.

       The  following  conventions  apply  to the SYNOPSIS section and can be used as a
       guide in other sections.

       bold text          type exactly as shown.
 Manual page man(1) line 1 (press h for help or q to quit)
```

4.

Navegação

cd change **d**irectory

- Muda o diretório atual
- Recebe como argumento o caminho para o qual será navegado
- Lembre-se que é possível utilizar os caracteres coringas

```
[pet@arch ~]$ cd Pictures/Wallpapers/
[pet@arch Wallpapers]$ cd ..
[pet@arch Pictures]$ cd ~
[pet@arch ~]$
```

ls list

- Imprime na tela o conteúdo (diretórios e arquivos) do diretório passado como argumento
- Caso não receba um argumento, utilizará o diretório atual

```
[pet@arch ~]$ ls
Documents  Pictures  Videos
[pet@arch ~]$ ls Pictures/Wallpapers/
wallpaper.png
[pet@arch ~]$
```

ls list

- Opções comuns:
  - `-a` → imprime arquivos e diretórios ocultos
  - `-l` → imprime informações do conteúdo
  - `-1` → imprime um item por linha
  - `-m` → separa os itens por vírgula
  - `-t` → ordena os arquivos em ordem decrescente de data

```
[pet@arch ~]$ ls
Documents  Pictures  Videos
[pet@arch ~]$ ls -lat
total 48
drwx------ 6 pet  pet  4096 Mar 19 20:07 .
-rw------- 1 pet  pet    42 Mar 19 20:07 .lesshst
drwxr-xr-x 7 pet  pet  4096 Mar 19 20:04 Documents
-rw------- 1 pet  pet   741 Mar 19 19:43 .viminfo
drwx------ 3 pet  pet  4096 Mar 19 19:43 .cache
-rw------- 1 pet  pet   401 Mar 19 19:12 .bash_history
drwxr-xr-x 2 pet  pet  4096 Mar 19 18:28 Videos
drwxr-xr-x 3 pet  pet  4096 Mar 19 18:23 Pictures
drwxr-xr-x 4 root root 4096 Mar 19 18:21 ..
-rw-r--r-- 1 pet  pet    21 Feb  2 03:38 .bash_logout
-rw-r--r-- 1 pet  pet    57 Feb  2 03:38 .bash_profile
-rw-r--r-- 1 pet  pet   172 Feb  2 03:38 .bashrc
[pet@arch ~]$
```

5.

Gerenciando arquivos e diretórios

touch

- Atualiza a data de última modificação e de último acesso do arquivo (sem modificar seu conteúdo)
- Caso não exista o arquivo especificado, cria um arquivo com o nome passado como argumento

```
[pet@arch ~]$ stat a.txt
  File: a.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 259,6   Inode: 16158665    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1001/     pet)   Gid: ( 1001/     pet)
Access: 2023-03-19 19:45:51.638879136 -0300
Modify: 2023-03-19 19:45:51.638879136 -0300
Change: 2023-03-19 19:45:51.638879136 -0300
 Birth: 2023-03-19 19:45:51.638879136 -0300
[pet@arch ~]$ touch a.txt
[pet@arch ~]$ stat a.txt
  File: a.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 259,6   Inode: 16158665    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1001/     pet)   Gid: ( 1001/     pet)
Access: 2023-03-19 19:47:15.708875809 -0300
Modify: 2023-03-19 19:47:15.708875809 -0300
Change: 2023-03-19 19:47:15.708875809 -0300
 Birth: 2023-03-19 19:45:51.638879136 -0300
```

mkdir make directory

- Cria um diretório usando o argumento passado como nome
- Caso já exista um diretório com o nome, imprime mensagem de erro

```
[pet@arch Documents]$ ls
[pet@arch Documents]$ mkdir UFPR
[pet@arch Documents]$ ls
UFPR
[pet@arch Documents]$
```

rm & rmdir remove & remove directory

- **rm** exclui arquivos a partir dos argumentos
- **rmdir** exclui diretório se este estiver vazio, caso contrário imprime mensagem de erro
- Também é possível excluir diretórios com **rm**, usando a opção **-r**

```
[pet@arch Documents]$ ls UFPR/
a.txt
[pet@arch Documents]$ rm UFPR/a.txt
[pet@arch Documents]$ rmdir UFPR/
[pet@arch Documents]$ ls
[pet@arch Documents]$
```

Nunca digite

```
sudo rm -rf /
```

Caractere de escape

- Existem caracteres especiais da shell, como o / e o caractere de espaço, que separa textos
- Assim, ao criar arquivos ou diretórios com o caracteres especiais, é preciso “escapá-los”
- Para isso, utilize a barra invertida “\” antes do caractere

```
[pet@arch Documents]$ mkdir Aulas Linux
[pet@arch Documents]$ ls
Aulas  Linux
[pet@arch Documents]$ rmdir *
[pet@arch Documents]$ mkdir Aulas\ Linux
[pet@arch Documents]$ ls
'Aulas Linux'
[pet@arch Documents]$
```

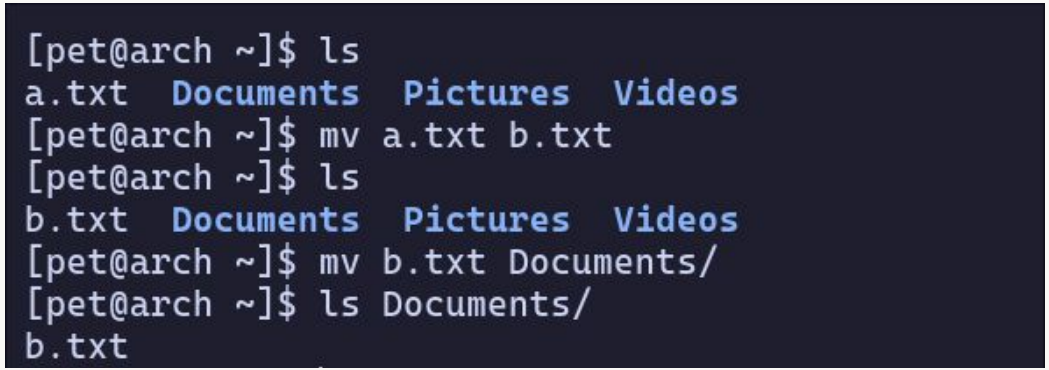
cp copy

- Copia arquivos e diretórios (com `-r`)
- `$ cp [arquivo] [destino]`
- `$ cp -r [diretório] [destino]`
- Caso o fim do destino seja um diretório, a cópia terá o mesmo nome
- Você pode terminar o destino com um arquivo, que será o nome da cópia

```
[pet@arch ~]$ cp a.txt Documents/
[pet@arch ~]$ cp a.txt b.txt
[pet@arch ~]$ cp -r Documents/ DocumentsBackup
[pet@arch ~]$ ls
a.txt  b.txt  Documents  DocumentsBackup  Pictures  Videos
[pet@arch ~]$
```

mv move

- Move (recorta) ou renomeia arquivos e diretórios
- `$ mv [arquivo/diretório] [destino]`
- `$ mv [arquivo/diretório] [novo nome]`
- Caso o destino seja o mesmo diretório da fonte, é possível alterar o nome do item



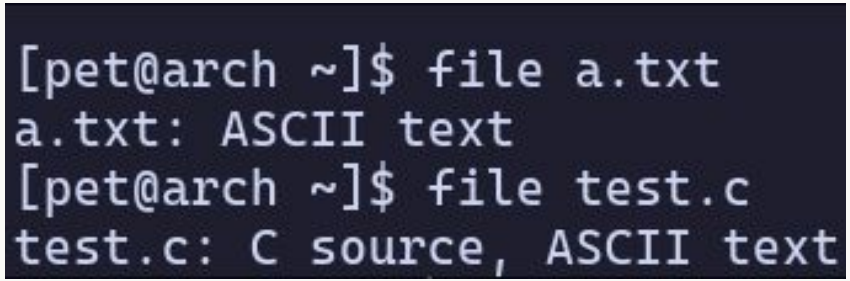
A opção `-i` em comandos como mv, cp e rm pedem confirmação de execução caso haja necessidade de sobrescrever um item

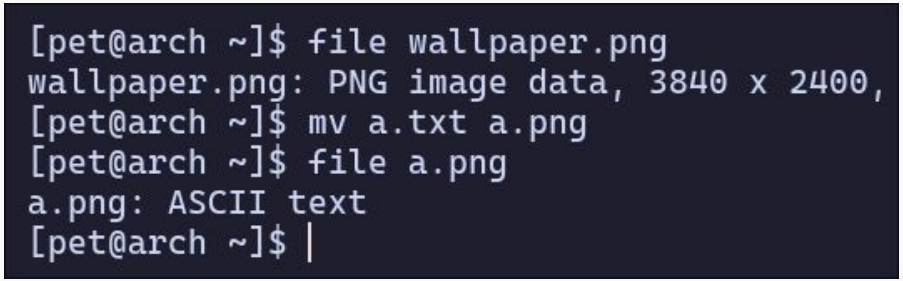
6.

Extraindo informações

file

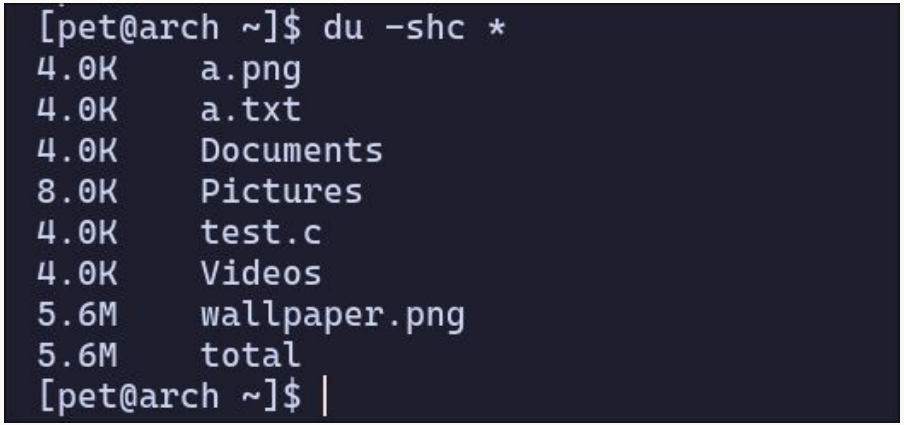
- Comando para imprimir o tipo do arquivo
- A princípio, parece um comando bobo, visto que há as extensões de arquivo
- Contudo, no Linux, o tipo do arquivo não se dá pela extensão, mas sim por **metadados**





du disk usage

- Imprime o tamanho em bytes de arquivo/diretório
- `$ du -shc *`



quota

- Quando logado na sua conta em um PC do DInf ou na macalan, utilize este comando para saber o consumo de sua quota
- `$ quota -s`

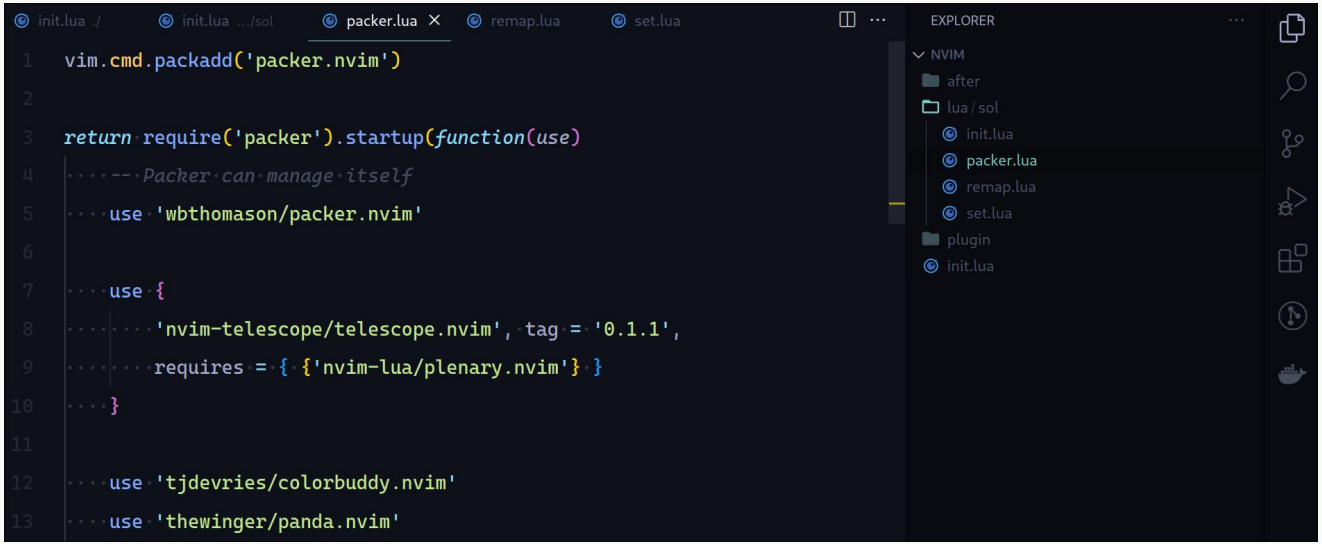
```
dlpg21@macalan:~$ quota -s
Disk quotas for user dlpg21 (uid 3603):
     Filesystem   space   quota   limit   grace   files   quota   limit   grace
urquell.home:/home
                  2309M   7813M   8008M           23444   40000   40000
urquell.home:/nobackup
                  2578M   7813M   8008M            6374       0       0
```

7.

Editores de texto

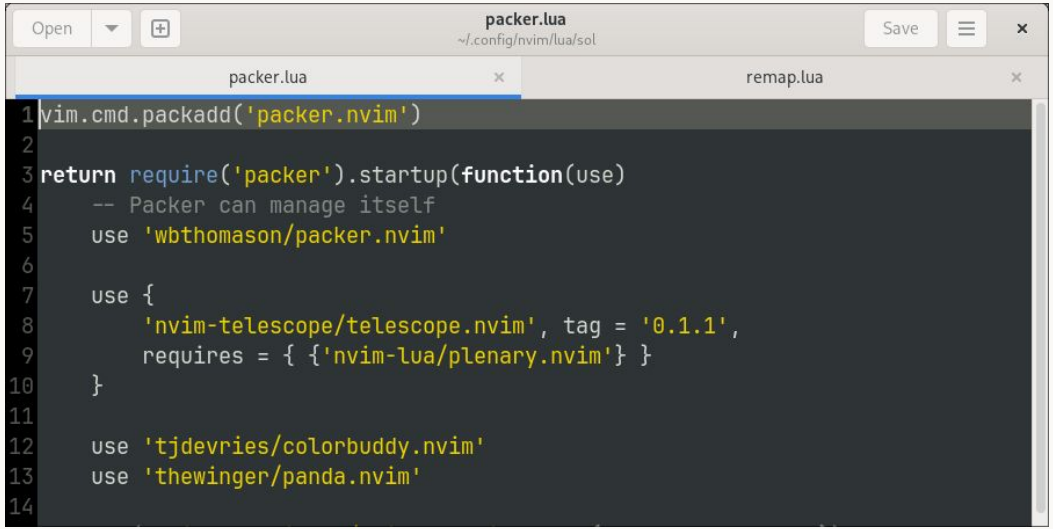
Visual Studio Code

- Criado pela Microsoft com o Electron, amplamente utilizado para desenvolvimento
- Disponibiliza inúmeros plugins feitos pela comunidade
- Há versões livres open-source, como o vscodium



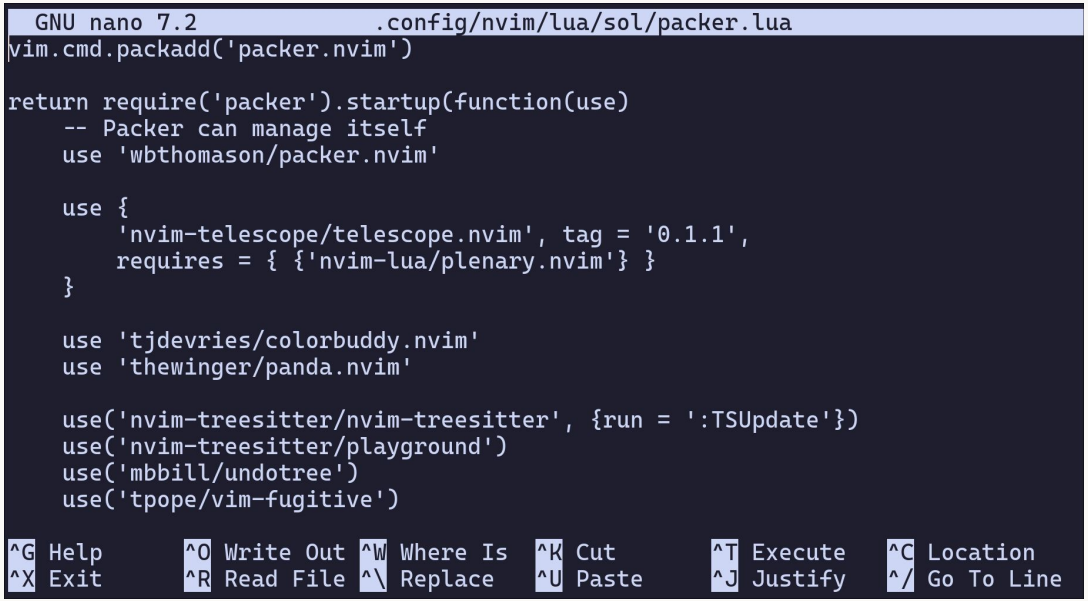
gedit

- Editor de texto com GUI do GNOME
- Simples, mas disponibiliza alguns plugins



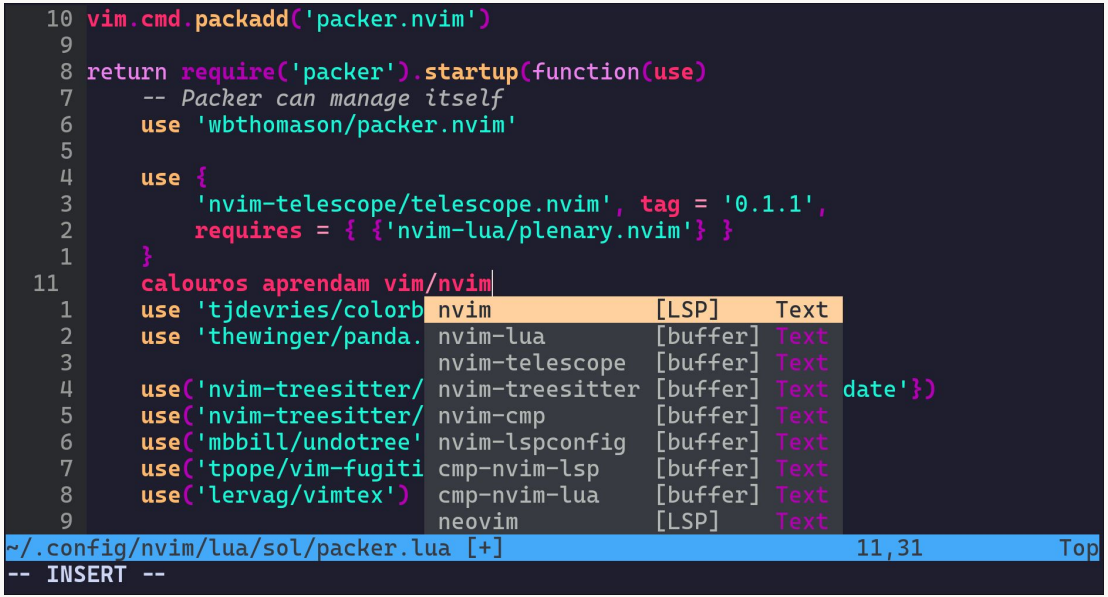
nano

- Editor de texto do GNU para terminais
- Interface simples, fácil aprendizado



vi, vim e neovim

- **vi** é um editor de texto para terminal criado para sistemas Unix
- **vim** foi lançado como uma versão extendida do **vi**, adicionando novas funcionalidades
- Posteriormente, **neovim** foi criado a partir do **vim**, melhorando a implementação de plugins



vi, vim e neovim

- 💡 Pesquise sobre o vim:
  - vimschool.netlify.app
  - openvim.com
  - tutorialspoint.com/vim/index.htm

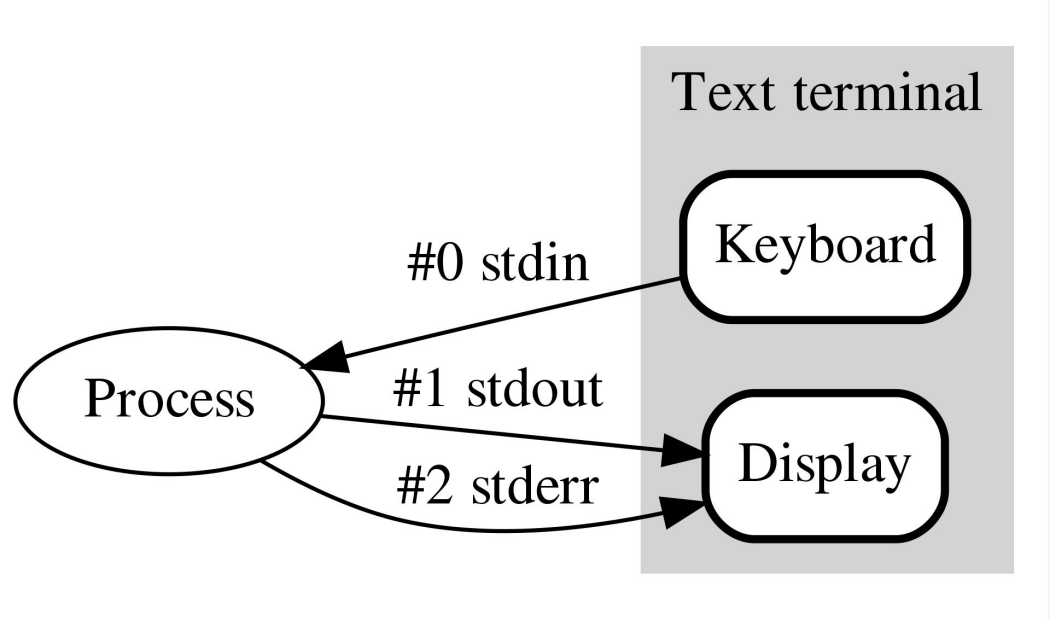
8.

Entrada e saída de dados

Streams

- Nos sistemas Unix, há a convenção de utilização de entradas e saídas padrão para a entrada e saída de dados
- São definidas três conexões/*streams*/fluxos padrão:
  - *stdin* → **st**an**d**ard **in**put (entrada padrão)
  - *stdout* → **st**an**d**ard **out**put (saída padrão)
  - *stderr* → **st**an**d**ard **err**or (erro padrão)
- Por estas streams ocorre o fluxo dados entre o teclado, processos e a tela, mediados pelo shell
- Por padrão, o *stdin* recebe dados do teclado, e *stdout* e *stderr* imprime mensagens na tela
- Contudo, é possível utilizar conteúdo de arquivos no *stdin*, e redirecionar a saída de *stdout* e *stderr* para arquivos, em vez da tela
- Então, a partir daqui, em vez de usarmos a expressão “imprimir na tela”, falaremos “escrever na saída/erro padrão”
- Veremos como redirecionar as *streams* adiante

Streams


Text terminal
Keyboard
#0 stdin
Process
#1 stdout
Display
#2 stderr

echo

- Escreve o texto (argumento) na saída padrão
- Por padrão, escreve o caractere de nova linha `\n` no fim do texto
- Para evitar a escrita do *new line*, utilize a opção **-n**

```
[pet@arch ~]$ echo 'Hello, world!'
Hello, world!
[pet@arch ~]$ echo -n 'Hello, world!'
Hello, world![pet@arch ~]$
```

cat & **tac** con**cat**enate

- Lê arquivo(s) passado(s) como argumento(s), o(s) concatena e escreve na saída padrão
- Caso não seja passado nenhum argumento, lê os dados da entrada padrão (*stdin*)
- O comando **tac** performa a mesma tarefa do **cat**, porém, escreve as linhas na ordem reversa

```
[pet@arch ~]$ cat hello.txt
Hello, world!
[pet@arch ~]$ cat test.c
#include <stdio.h>

int main() {
    return 0;
}
[pet@arch ~]$ cat hello.txt test.c
Hello, world!
#include <stdio.h>

int main() {
    return 0;
}
[pet@arch ~]$
```

```
[pet@arch ~]$ tac hello.txt
Hello, world!
[pet@arch ~]$ tac test.c
}
    return 0;
int main() {

#include <stdio.h>
[pet@arch ~]$ tac hello.txt test.c
Hello, world!
}
    return 0;
int main() {

#include <stdio.h>
[pet@arch ~]$
```

9.

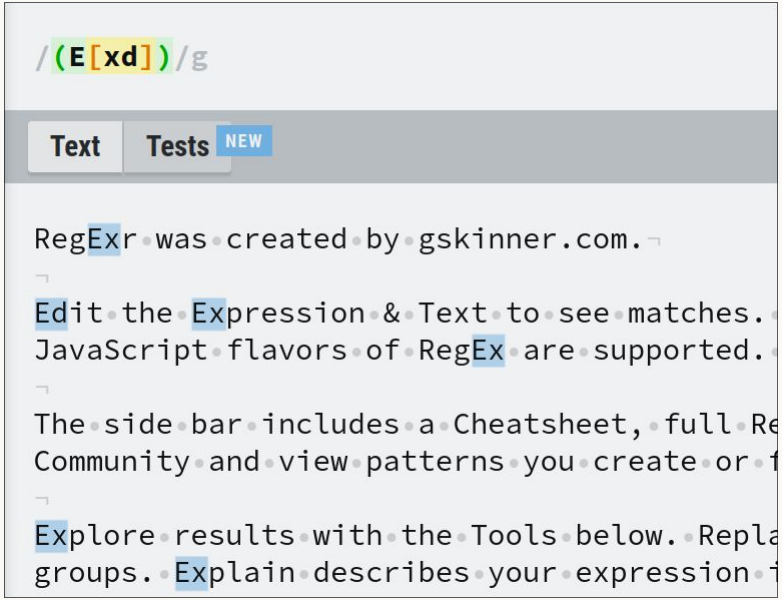
Caracteres coringas

Redirecionamento das *streams*

- É possível alterar a fonte da *stdin* e as saídas de *stdout* e *stderr* a partir de caracteres coringas
- Para redirecionar a saída padrão para um arquivo, utiliza-se **>** e **>>**, destrutivo e não-destrutivo respectivamente
  - `echo ‘Hello, world!’ > hello.txt`
  - `echo ‘abc’ >> hello.txt`
  - `echo ‘destruindo’ > hello.txt`
  - `echo ‘echo abc’ > echo.txt`
- Para enviar dados de um arquivo para *stdin*, utiliza-se **<**
  - `cat < hello.txt`
  - `bash < echo.txt`

Regex regular expressions

- As expressões regulares são amplamente utilizadas em diversas áreas da computação
- Com elas, é possível representar **padrões** de texto a partir de sequência de caracteres
- **Alguns** de seus caracteres são possíveis de serem usados no shell como caracteres coringas
- 💡 Pesquise sobre **regex**



regexr.com

Caracteres coringas

- **`*`** → representa zero ou mais ocorrências de quaisquer caracteres
- **`?`** → representa uma ocorrência de um caractere qualquer, pode ser combinado (ex: **`?????.txt`**)
- **`[]`** → representa uma ocorrência de algum caractere presente entre os colchetes, pode ser combinado (ex: **`[2468][13579].txt`**)
- **`{}`** → com chaves é possível representar sequências de caracteres
  - `echo {1..100}`
  - `echo {a..z}`
- & → colocado no fim de um comando, coloca o processo em *background* e libera o prompt
  - gedit teste.txt &

Caracteres coringas

- ; → com este caracter é possível separar comandos em apenas uma linha
  - `echo a; echo b`
- **$** → usada para representar variáveis de ambiente do shell
  - `echo $PATH`
- “ → usada para representar texto permitindo caracteres especiais
  - `echo “Data: $(date)”`
- ‘ → usada para representar texto anulando efeito dos caracteres especiais
  - `echo ‘Data: $(date)’`

Caracteres coringas

```
[pet@arch test]$ touch ts{1..5}.txt
[pet@arch test]$ ls
ts1.txt  ts2.txt  ts3.txt  ts4.txt  ts5.txt
[pet@arch test]$ touch tr{1..5}.txt
[pet@arch test]$ ls
tr1.txt  tr3.txt  tr5.txt  ts2.txt  ts4.txt
tr2.txt  tr4.txt  ts1.txt  ts3.txt  ts5.txt
[pet@arch test]$ ls tr*.txt
tr1.txt  tr2.txt  tr3.txt  tr4.txt  tr5.txt
[pet@arch test]$ ls t?[1-3].txt
tr1.txt  tr2.txt  tr3.txt  ts1.txt  ts2.txt  ts3.txt
```

Exercício

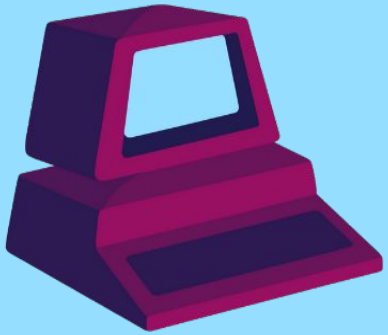
```
$ wget https://www.inf.ufpr.br/vm20/linux/aula2.tar.gz
$ tar -xvf aula2.tar.gz
```

Avalie a aula

forms.gle/VX45YdL5WJsp7zMy6

Conta como presença!

Obrigado!



PET
COMPUTAÇÃO

pet.inf.ufpr.br

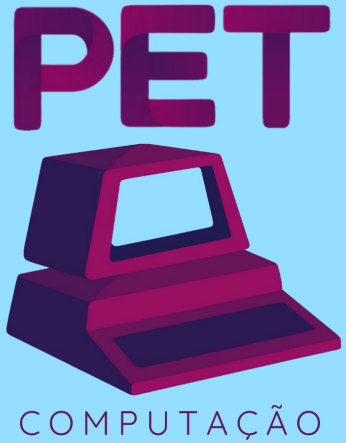
pet@inf.ufpr.br

@petcompufpr

Linux

Aula III

Login: petsemcalX
Senha: PetSemcal#X



2023

Relembrando...

Comandos

- `rm` - remove arquivo/diretório
- `mv [arquivo] [destino]` - move arquivo/diretório
- `mv [arquivo] [novo nome]` - renomeia arquivo/diretório
- `cp` - copia arquivo/diretório
- `file` - especifica tipo do arquivo
- `touch` - atualiza data de acesso ou cria novo arquivo
- `du` - mostra o tamanho de arquivo/diretório
- `cat` - concatena e imprime o conteúdo do arquivo

Texto para treino

wget https://www.inf.ufpr.br/vm20/linux/memorias.txt

1.

Usuários e grupos

Usuários e grupos

- Usuários e grupos são usados para controle de acesso a arquivos e diretórios
- Usuários podem ser adicionados e removidos, com eles é possível logar no sistema
- Grupos são conjuntos de usuários, que podem receber permissões específicas que se aplicam a todos os usuários presentes nele
- Ao instalar o Linux, um usuário é criado por padrão, o usuário **root**

Superusuário

- O root é um **superusuário**, ou seja, é um usuário administrador que pode controlar qualquer aspecto do sistema operacional, desde acessar qualquer arquivo/diretório a rodar qualquer comando
- Como usuário comum, é possível rodar um comando como root usando o comando **sudo** como prefixo
  - `$ sudo apt install nvim`
- Contudo, para poder utilizar o comando sudo, o usuário deve estar no grupo sudoers, que apenas o **root** e outros usuários sudoers conseguem gerenciar
  - `$ cat /etc/sudoers` (só conseguirá rodar como superusuário, rs)

2.

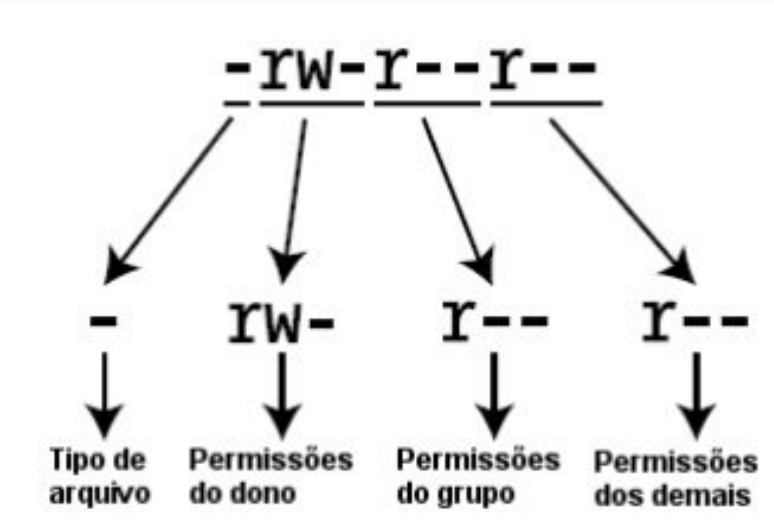
Permissões

Permissões

- Há três tipos de permissões aplicadas a arquivos e diretórios:
  - **R**ead: leitura
  - **W**rite: escrita, alteração e deleção
  - E**x**ecute: execução
- Estas permissões podem ser definidas para três entidades:
  - **U**ser: seu usuário
  - **G**roup: usuários presentes nos seus grupos
  - **O**thers: o resto
- Cada arquivo e diretório possuem um **dono**, que é o criador do item
- O dono pode gerenciar seus arquivos e diretórios para atribuir permissões de leitura, escrita e/ou execução a outros usuários
- O superusuário tem permissão para leitura, escrita e execução em qualquer arquivo/diretório!

Permissões

- Rode: `$ ls -l`




```
$ ls -l
drwxr-xr-x 2 chapolin pet-user 4096 2011-02-25 16:22 Comandos
-rw-r--r-- 1 chapolin pet-user  392 2011-02-25 14:37 Makefile
```

chmod change mode

- Para alterar permissões, utiliza-se o comando chmod
- O primeiro argumento são as entidades em que as permissões serão mudadas
- O segundo argumento é o tipo de alteração
- O terceiro argumento é o tipo de permissão
- A notação textual segue:
  - `$ chmod [ugoa] [+-=] [rwx] [arquivo/diretório]`
- As possibilidades do terceiro argumento podem ser combinadas
  - **Ex:** `$ chmod ug=rw, o-rw hello.txt`
- Ao ser utilizado em diretórios, é possível utilizar a opção -R (**R**ecursivo) para aplicar as mudanças em todos os arquivos/diretórios filhos
- 💡 Pesquise sobre a notação octal do **chmod**

public_html

- Ao adicionar um arquivo no seu diretório public_html do DInf, não se esqueça de alterar as permissões para permitir a leitura e execução do arquivo pelo servidor nginx
- `$ chmod o+rx index.html`

3.

Histórico

history

- A shell, por padrão, escreve os comandos executados num arquivo de histórico
- Bash: ~/.bash_history
- zsh: ~/.zsh_history
- Assim, é possível utilizar o comando **history** que exibe o histórico dos comandos

4.

Redirecionamento de streams entre comandos

Pipeline

- Você pode redirecionar a saída de um comando (*stdout*) para ser usada como entrada (*stdin*) de outro usando **pipe** (caractere '|')
- `comando1 | comando2 | comando3` - comando1 gera entrada pro comando2 e o comando2 pro comando3.
- Exemplos:
  - `ls | tac` - imprime o **ls** da última linha para a primeira
  - `ls -t | tac` - imprime o **ls** em ordem crescente de data

5.

Mais comandos de leitura

head

- Por padrão, lê as **primeiras** 10 linhas de um arquivo
- É possível especificar quantas linhas lidas com a opção `-n [número]`
- Ex:
  - `$ head arquivo.txt`
  - `$ head -n 20 arquivo.txt`

tail

- Por padrão, lê as **últimas** 10 linhas de um arquivo
- É possível especificar quantas linhas lidas com a opção `-n [número]`
- Ex:
  - `$ tail arquivo.txt`
  - `$ tail -n 20 arquivo.txt`

less

- Lê um arquivo de forma interativa
- Isto é, não imprime todo o arquivo na tela, em vez disso, é possível navegar pelo arquivo utilizando setas, mouse etc.
- Ideal para arquivos grandes
- “q” para sair, “h” para mostrar a tela de ajuda
- “/padrao” grifa as ocorrências de “padrao” no documento
- Versão melhorada do comando “more”

6.

Buscando arquivos

find

- Formato: **find** `[ponto-inicial] [opções] [argumento]`
- Ponto inicial: por onde começar a procura (padrão: diretório atual)
- Opções: no geral, especifica que tipo de argumento o comando irá procurar, como nome, data de criação, permissão. Exemplo de opção: -iname.
- Argumento: o que o comando irá procurar levando em consideração o tipo dado nas opções.

7.

Filtrando texto

grep

- Procura por uma palavra/frase/expressão dentro de um arquivo.
  - **G**lobal **R**egular **E**xpression **P**rint
  - **G**lobally looks for a **R**egular **E**xpression and **P**rint
  - **G**lobal **R**egular **E**xpression **P**arser
- Formato: **grep** `[opções][expressão][arquivo]`
- `cat doc.txt | grep legal -` Pega as linhas de **doc.txt** com a palavra “legal”
- Você pode sempre usar o man para ver as opções:
  - `man grep`

cut

- Usado para “cortar” o texto por colunas
- Opções
  - `-d'c'`: usa ‘**c**’ como caractere delimitador
  - `-fn` : retorna apenas o campo (*field*) número **n**
  - `-fn,m` : retorna apenas os campos (*field*) número **n e m**
- O caractere TAB é o padrão de delimitador
- Parecido com outro comando: **awk**
- `cat documento.csv | cut -d',' -f1 -` Colunas separadas por vírgula, coluna 1

Exercício

```
$ wget https://www.inf.ufpr.br/vm20/linux/aula2.tar.gz
$ tar -xvf aula2.tar.gz
```

Exercício

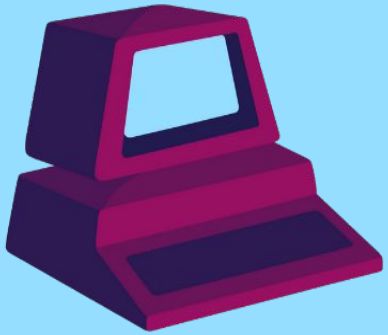
```
$ wget https://www.inf.ufpr.br/vm20/linux/aula2.tar.gz
$ tar -xvf aula2.tar.gz

$ wget https://www.inf.ufpr.br/vm20/linux/aula3.tar.gz
$ tar -xvf aula3.tar.gz
```

Avalie a aula

forms.gle/CnxErdMxXF7TTAmT9

Conta como presença!

Obrigado!



PET
COMPUTAÇÃO

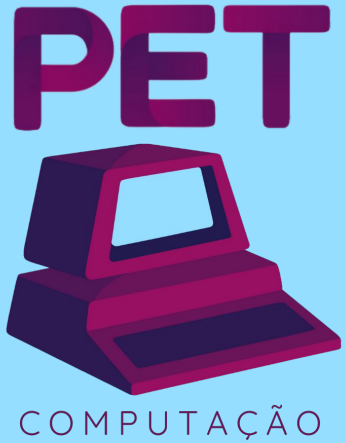
pet.inf.ufpr.br

pet@inf.ufpr.br

@petcompufpr

Linux

Aula IV



Login: petsemcalX
Senha: PetSemcal#X

2023

Relembrando...

Comandos

- `touch` - atualiza data de acesso ou cria novo arquivo
- `rm` - remove arquivo/diretório
- `mv [arquivo] [destino]` - move arquivo/diretório
- `man` - mostra o manual do comando
- `cp` - copia arquivo/diretório
- `less` - lê o arquivo de forma interativa
- `cat` - concatena e imprime o conteúdo do arquivo
- `sudo` - usa os poderes do super usuário

Exercício surpresa!!

- O que estes comandos fazem?
  - `man htop`
  - `rsync -h`
  - `wget --help`
  - `touch {1..5}`

1.

Recomendações de programas

Navegadores Web

Firefox

Chromium

Qutebrowser

Multimídia

VLC
Vídeos e músicas

Eye of GNOME
Imagens

sxiv
Imagens

Gerenciadores de arquivos

LF

Terminal

Files for GNOME
Gráfico

Nemo
Gráfico

Office documentos, planilhas e apresentações

LibreOffice

WPS

Google Docs

Pesquise sobre o Latex!

Lista de aplicações

wiki.archlinux.org/title/List_of_applications

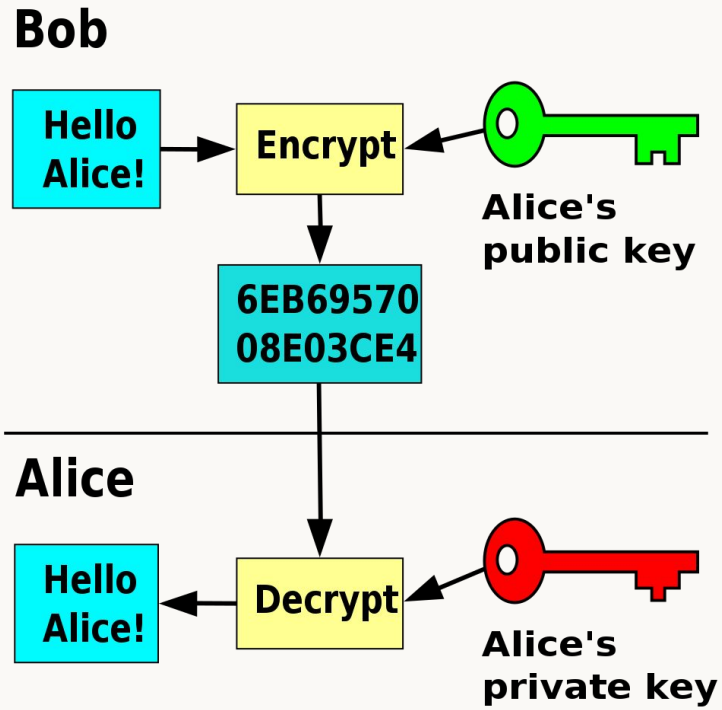
2.

Secure Shell

Secure Shell

- O *Secure Shell* é um protocolo de rede criptografado, sendo seguro para utilizar em qualquer rede (segura ou não)
- É um protocolo TCP/IP na camada de aplicação, i.e., a mais alta
- Com o **ssh**, é possível fazer login em uma máquina remota, assim, tendo acesso a um shell naquele computador
- Como criptografia, utiliza a Criptografia de Chave Pública
- Além de abrir um shell em máquinas remotas, é possível, também, mover arquivos entre os computadores
- 💡 Pesquise sobre o OpenSSH e a criptografia de chave pública

Secure Shell



ssh Secure Shell

- Conecta seu shell a outra máquina de forma remota
  - comando: `ssh <user>@<endereço>`
  - conecta na porta 22

```
fontoura@nerv:~$ ssh vfa20@ssh.inf.ufpr.br
```

- As principais máquinas do DInf
  - macalan - `servidora de uso geral`
  - orval - `servidora para processamento que utiliza muitos recursos`
  - cpu1 e cpu2 - `servidoras para processamento genérico`

SCP

- Copia um arquivo local para uma máquina remota
  - comando: `scp <arquivo> <user>@<endereço>:<caminho>`
- Copia um arquivo remoto para a máquina local
  - comando: `scp <user>@<endereço>:<caminhoDoArquivo> <destinoLocal>`
- Busque usar caminhos absolutos ao executar o `scp`, vai te evitar problemas

Exercício surpresa!!

- O que estes comandos fazem?
  - `sudo apt install curl`
  - `chmod g+r -R <dir>/`
  - `rm {1..10}`
  - `tar -czvf <dir>.tar.gz <dir>/`

3.

sshfs

mount & umount

- Monta dispositivos (USB, HD externo, etc.) em um diretório para poder acessar os arquivos
- Lembre-se, os dispositivos são representados como arquivos em /dev
- Ex:
  - `mkdir pendrive && mount /dev/sda1 pendrive`
- Geralmente, os SOs montam automaticamente
- Após a utilização, por segurança, deve-se desmontar o dispositivo
- Ex:
  - `umount pendrive && rmdir pendrive`

sshfs

- Monta diretórios remotos de outras máquinas na máquina local
  - comando: `sshfs <user>@<endereço>:<caminhoRemoto> <diretórioAlvo>`

- Use `fusermount` para desmontar o diretório
  - comando: `fusermount -u <diretório>`

4.

Comandos legais do ssh

Comandos utilizados durante o ssh

- `whoami` - mostra o nome do usuário no sistema
- `hostname` - mostra o nome do sistema
- `who` - mostra quem mais está conectado no sistema
- `finger` - mostra os detalhes de um usuário no sistema

Exercício surpresa!!

- O que estes comandos fazem?
  - `rm ./*.csv`
  - `ls -l | grep ^d | awk '{print $NF}'`
  - `less <arquivoDeTexto>`
  - `du -hc <dir>`

5.

Exercício em sala

sshfs

- Dê SCP na máquina do seu colega ao lado e tente copiar algum arquivo dela
- Crie um arquivo e copie ele para a máquina do seu colega usando SCP

Exercício

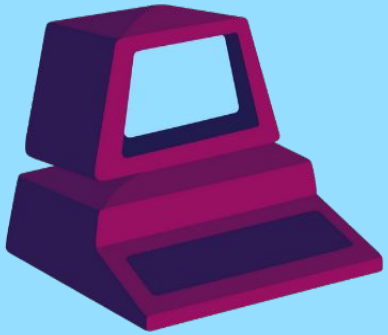
```
$ wget https://www.inf.ufpr.br/vm20/linux/misterio.tar.gz
$ tar -xvf misterio.tar.gz
```

Avalie a aula

forms.gle/Xks4K5QCwvR8nsvs5

Conta como presença!

Obrigado!



PET
COMPUTAÇÃO

pet.inf.ufpr.br

pet@inf.ufpr.br

@petcompufpr