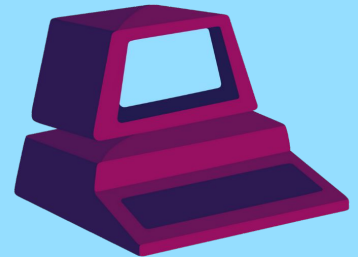


Linux

Aula II

Login:
Senha:

PET



COMPUTAÇÃO

2026

Login e senha para o curso



Criamos contas temporárias para vocês usarem no Curso de Linux.

Observe que o computador que você está usando possui um adesivo com uma letra e um número. Exemplo: H30 ou i12.

Você vai usar esse número para o seu login e senha.

Login e senha para o curso

Computadores do LAB 12:

- Considere @ = número do seu computador.
- Exemplo: se o seu computador é o H30, @ = 30.

Computadores do LAB 3:

- Considere @ = número do seu computador + 60.
- Exemplo: se o seu computador é o i12, @ = 72.

Computadores do LAB 4:

- Considere @ = número do seu computador + 80.
- Exemplo: se o seu computador é o l23, @ = 103.

Login e senha para o curso

Sabendo o seu @, seu login e senha são:

- Login: clinux@
- Senha: clinux@#@

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 (parecido com 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.

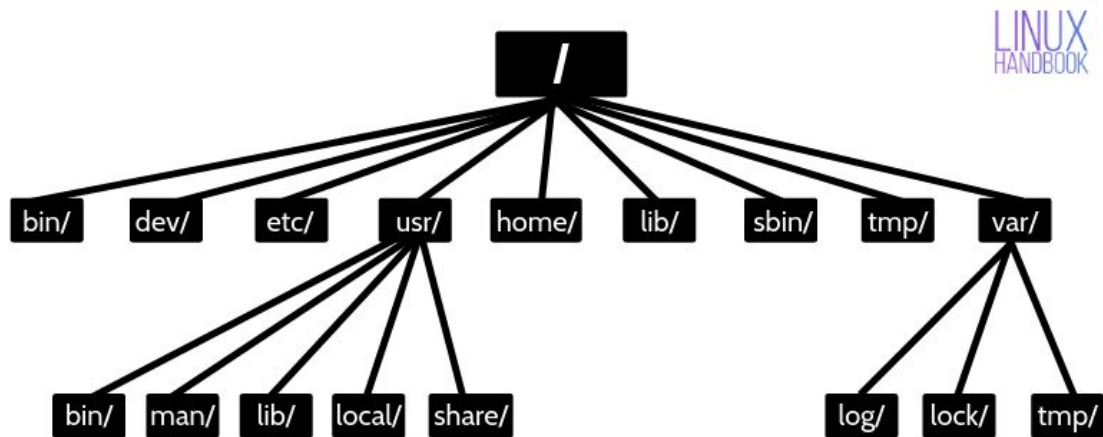
Sistema de Arquivos



Sistema de arquivos

- Algumas características de um Sistema de Arquivos são:
 - Realiza o gerenciamento do espaço de armazenamento do disco
 - Organização do armazenamento (disco) em arquivos contidos em diretórios (pastas)
 - Cada diretório pode conter arquivos e outros diretórios
 - Diretórios e arquivos possuem nome (*case-sensitive*)
 - Links simbólicos (atalhos)
 - Navega entre diretórios

Sistema de archivos



Sistema de arquivos não precisa decorar!

- / → **diretório raiz** (*root*), todos diretórios e arquivos do SO estão contidos nele
- /bin → arquivos binários executáveis (**os programas**), como comandos da shell
- /dev → arquivos virtuais 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
- /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.)
- /sbin → binários do sistemas, só podem ser executados pelo **root**

Working directory

- Ao iniciar um shell, 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)
- O **diretório atual** também é representado através de um **ponto “.”**

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 “/”)
- Ex.:
/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 **depende do *working directory***
- ./Pictures/Wallpapers/wallpaper.png
(assumindo que o diretório atual é /home/pet)

3.

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`

*Não precisa se assustar

whoami

- Imprime na tela o usuário atual

```
[pet@arch ~]$ whoami  
pet  
[pet@arch ~]$ |
```

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)

```
[david:~]$ tty  
/dev/pts/3  
[david:~]$
```

```
[david:~]$ tty  
/dev/pts/1  
[david:~]$
```

```
[david:~]$ tty  
/dev/pts/2  
[david:~]$
```

```
[david:~]$ tty  
/dev/pts/4  
[david:~]$
```

who

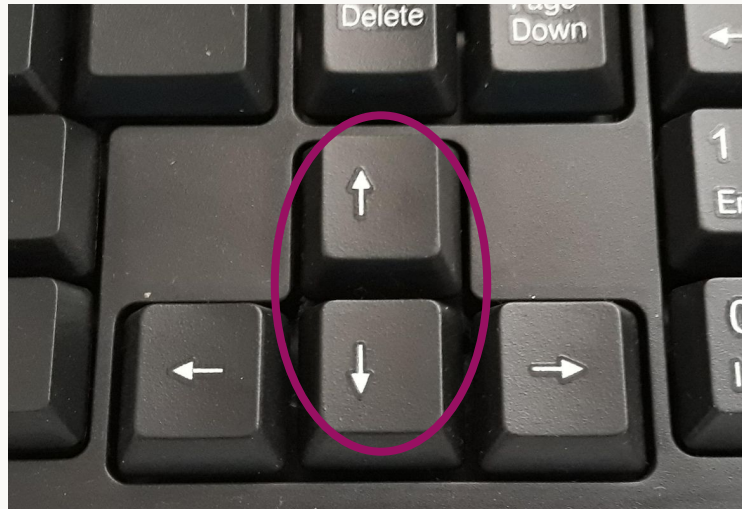
- Imprime na tela os usuários conectados ao sistema

```
[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:~$ |
```

Histórico

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



4.

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, CONFORMING 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)
```

5.

Navegação



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

cd change directory

- 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 ~]$ |
```

6.

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
```

Exercício surpresa!!

- O que estes comandos fazem?
 - `man htop`
 - `rsync -h`
 - `wget --help`
 - `cd /`
 - `cd ~/../..`

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 os 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

```
[pet@arch ~]$ ls
a.txt Documents Pictures Videos
[pet@arch ~]$ mv a.txt b.txt
[pet@arch ~]$ ls
b.txt Documents Pictures Videos
[pet@arch ~]$ mv b.txt Documents/
[pet@arch ~]$ ls Documents/
b.txt
```

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

7.

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**

```
[pet@arch ~]$ file a.txt
a.txt: ASCII text
[pet@arch ~]$ file test.c
test.c: C source, ASCII text
```

```
[pet@arch ~]$ file wallpaper.png
wallpaper.png: PNG image data, 3840 x 2400,
[pet@arch ~]$ mv a.txt a.png
[pet@arch ~]$ file a.png
a.png: ASCII text
[pet@arch ~]$ |
```

du disk usage

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

```
[pet@arch ~]$ du -shc *
4.0K    a.png
4.0K    a.txt
4.0K    Documents
8.0K    Pictures
4.0K    test.c
4.0K    Videos
5.6M    wallpaper.png
5.6M    total
[pet@arch ~]$ |
```

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
```

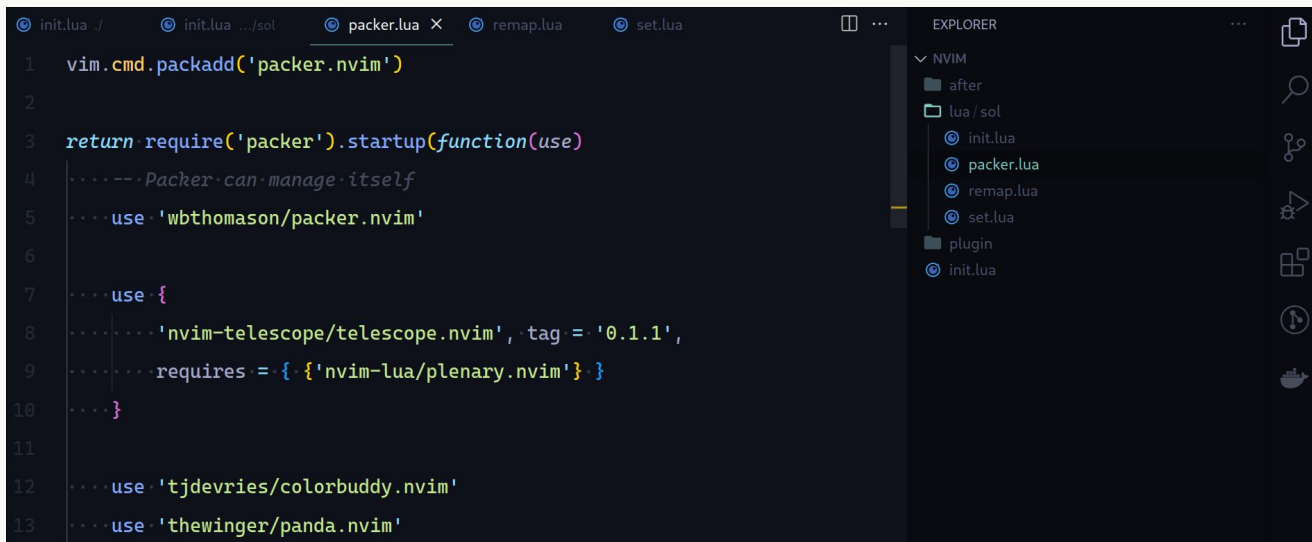
8.

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 vsodium

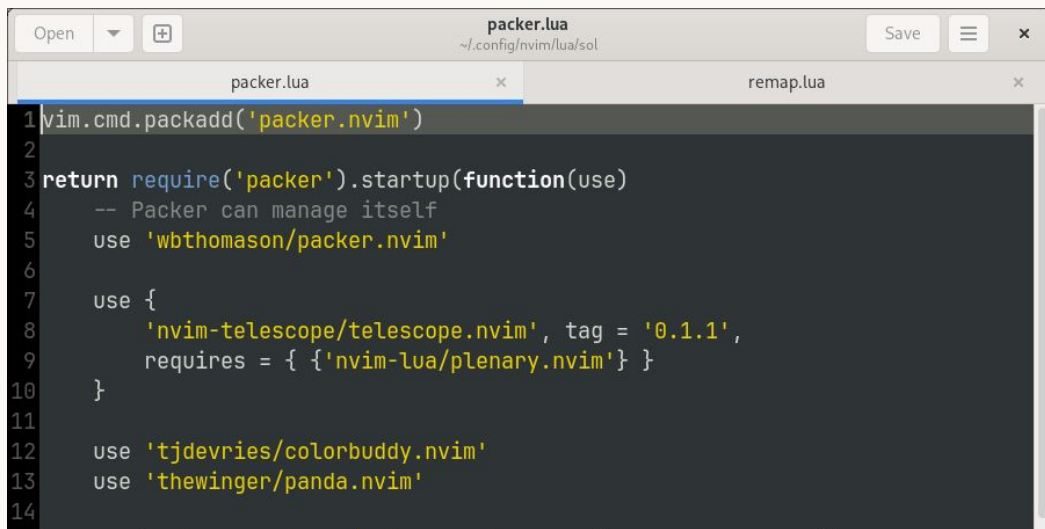


The screenshot shows the Visual Studio Code interface with a dark theme. The main editor window displays a Lua script for managing Nvim plugins. The script includes a packadd command and a startup function that uses the packer plugin manager to install several plugins, including telescope, plenary, colorbuddy, and panda. The Explorer sidebar on the right shows the file structure of the project, including a lua/sol directory with init.lua, packer.lua, remap.lua, and set.lua, and a plugin directory with init.lua.

```
1 vim.cmd.packadd('packer.nvim')
2
3 return require('packer').startup(function(use)
4   ...--Packer can manage itself
5   ... use 'wbthomason/packer.nvim'
6
7   ... use {
8     ... 'nvim-telescope/telescope.nvim', tag = '0.1.1',
9     ... requires = { {'nvim-lua/plenary.nvim'} }
10  ... }
11
12  ... use 'tjdevries/colorbuddy.nvim'
13  ... use 'thewinger/panda.nvim'
```

gedit

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



The screenshot shows the gedit text editor interface. The title bar indicates the file is 'packer.lua' located at '~/.config/nvim/lua/sol'. The editor has two tabs open: 'packer.lua' and 'remap.lua'. The main editing area contains the following Lua code:

```
1 vim.cmd.packadd('packer.nvim')
2
3 return require('packer').startup(function(use)
4   -- Packer can manage itself
5   use 'wbthomason/packer.nvim'
6
7   use {
8     'nvim-telescope/telescope.nvim', tag = '0.1.1',
9     requires = { {'nvim-lua/plenary.nvim'} }
10  }
11
12  use 'tjdevries/colorbuddy.nvim'
13  use 'thewinger/panda.nvim'
14
```

nano

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

```
GNU nano 7.2 .config/nvim/lua/sol/packer.lua
vim.cmd.packadd('packer.nvim')

return require('packer').startup(function(use)
  -- Packer can manage itself
  use 'wbthomason/packer.nvim'

  use {
    'nvim-telescope/telescope.nvim', tag = '0.1.1',
    requires = { {'nvim-lua/plenary.nvim'} }
  }

  use 'tjdevries/colorbuddy.nvim'
  use 'thewinger/panda.nvim'

  use('nvim-treesitter/nvim-treesitter', {run = ':TSUpdate'})
  use('nvim-treesitter/playground')
  use('mbbill/undotree')
  use('tpope/vim-fugitive')

```


^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^\ Replace	^U Paste	^J Justify	^/ Go To Line

vi, vim e neovim

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

```
10 vim.cmd packadd('packer.nvim')
9
8 return require('packer').startup(function(use)
7   -- Packer can manage itself
6   use 'wbthomason/packer.nvim'
5
4   use {
3     'nvim-telescope/telescope.nvim', tag = '0.1.1',
2     requires = { {'nvim-lua/plenary.nvim'} }
1   }
11 calouros aprendam vim/nvim
1 use 'tjdevries/colorb nvim [LSP] Text
2 use 'thewinger/panda. nvim-lua [buffer] Text
3 nvim-telescope [buffer] Text
4 use('nvim-treesitter/ nvim-treesitter [buffer] Text date'})
5 use('nvim-treesitter/ nvim-cmp [buffer] Text
6 use('mbbill/undotree' nvim-lspconfig [buffer] Text
7 use('tpope/vim-fugiti cmp-nvim-lsp [buffer] Text
8 use('lervag/vimtex') cmp-nvim-lua [buffer] Text
9 neovim [LSP] Text
~/./config/nvim/lua/sol/packer.lua [+] 11,31 Top
-- INSERT --
```

vi, vim e neovim

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

Exercício

```
$ wget https://www.inf.ufpr.br/dlpg21/linux/aula2.tar.gz
```

```
$ tar -xvf aula2.tar.gz
```

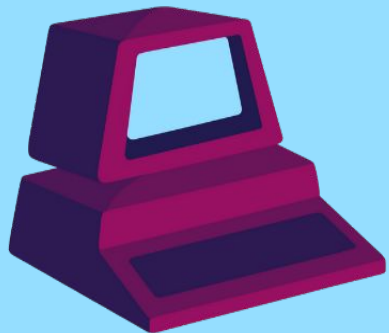
Avalie a aula

forms.gle/Yhti3bd3j5Uuz3CLA



Conta como presença!

Obrigado!



PET
COMPUTAÇÃO

pet.inf.ufpr.br
pet@inf.ufpr.br
[@petcompufpr](https://twitter.com/petcompufpr)