# Comandos - Linux

☑ Reviewed ☐

▼ ***Teclas Úteis | Sistema de Arquivos | Desligar***

## Introdução ao Bash

### Aviso de Comando (Prompt)

- `SHIFT` e pressionando `PGUP` ou `PGDOWN`: Isso é útil para ver textos que rolaram rapidamente para cima.
- `du -hs`: Mostrará o tamanho total (pode usar o -a (espaço ocupado por todos os arquivos)
- `bash - version`

### Teclas Úteis

- `Back Space ("<--"):` Apaga um caractere à esquerda do cursor.
- `Del`: Apaga o caractere acima do cursor.
- `CTRL+A:` Move o cursor para o início da linha de comandos.
- `CTRL+E`: Move o cursor para o fim da linha de comandos.
- `CTRL+U`: Apaga o que estiver à esquerda do cursor. O conteúdo apagado é copiado para uso com `CTRL+y.`
- `CTRL+K:` Apaga o que estiver à direita do cursor. O conteúdo apagado é copiado para uso com CTRL+y.
- `CTRL+L`: Limpa a tela e mantém o texto que estiver sendo digitado na linha de comando (parecido com o comando clear).
- `CTRL+Y:` Coloca o texto que foi apagado na posição atual do cursor.

## Comandos

- comando [opções] [argumentos]
  - [opções]: Modifica o comportamento de um comando.
  - [argumento]: Especifica algo para o qual o comando agirá (como nome de arquivo ou nome de usuário).
  - Exemplo: ls -lha

## Padrão de Hierarquia do Sistema de Arquivos (FHS)

1. **/bin**: Binários do sistema (*TUDO* que é executável para o Linux).

   Ex: ls, vim, login, exit, cd, users.

2. **/boot:** Inicialização do sistema, onde tem o kernel linux etc.

   Ex: grub (se é linux ou windows), vmlinuz (kernel do linux, que varia de 5,1M).

3. **/run:** Tudo que está rodando desde o último boot. É um arquivo temporário.

   Exemplo: network, daemons, users.

4. **/media**: Mídia removível que já está montada automaticamente.

5. **/dev:** Tudo que é dispositivo. Tudo no linux é um arquivo, inclusive um HD, SATA (sda5), etc.

   Ex: sda (primeiro disco), sdb (segundo disco), sda5 (onde pode dividir em partições). Temos 2 tipos de dispositivos: Temos dispositivos de bloco (armazém. - HD, PENDRIVE) e caractere (transferindo informação - Primeira porta serial, terminais de texto, um dispositivo que descarta todos os dados escritos nele, dispositivos Random, mouse).

   - Exemplo: brw-rw---- (bloco), crw-rw---- (caractere).

6. **/share:** Documentação/ manual/help.

7. **/etc:** Todas as configurações da máquina ou arquivos de configuração.

   Ex: nome da máquina (hostname), parte de rede para configuração, log motd, samba, nginx, apache, etc.

8. **/home:** Contém o diretório de cada usuário e as coisas de cada um.

Ex: /home/john/ (pasta pessoal do usuário "John").

9. **/lib**: Bibliotecas dinâmicas compartilhadas entre os programas.

   Ex: /lib/libc.so.6 (biblioteca C compartilhada).

10. **/lost+found:** Usado para armazenar arquivos ou diretórios recuperados pelo utilitário fsck em caso de problemas no sistema de arquivos.

    Ex: /lost+found/recovered_file.txt (arquivo recuperado), libdiscover.so.2.0.1, módulos, drivers (módulos de kernel com algumas extensões .ko).

11. **/mnt:** Ponto de montagem temporário para dispositivos e sistemas de arquivos adicionais.

    Ex: /mnt/external_hd/ (disco rígido externo montado).

12. **/opt:** Ponto onde instalei o SO e arquivos de terceiros que não são do linux. Pouco utilizado, porque existe gerenciamento de pacotes (como apt, yum, dnf, etc.) que facilitam a instalação e a atualização de software.

    Ex: /opt/google/chrome/ (diretório de instalação do Google Chrome), Aplicativos de terceiros com requisitos específicos de instalação. Software personalizado ou de desenvolvedor que não faz parte dos repositórios de pacotes da distribuição.

13. **/proc:** Um sistema de arquivos virtual que fornece informações sobre o estado do kernel e processos em execução, informações dinâmicas sobre o sistema. Todo processo que está executando está no /proc.

    Ex: cpuinfo, meminfo(total de memória), uptime.

14. **/root:** O diretório pessoal do usuário "root" (superusuário) do sistema.

    Ex: /root/ (pasta pessoal do superusuário).

15. **/sbin:** Contém programas usados pelo superusuário (root) para administração e controle do sistema.

    Ex: add usuário, deletar, modificar, swapon e off, shutdown, runuser.

16. **/sys:** Similar ao /proc, é um sistema de arquivos virtual usado para interagir com configurações e informações do kernel.

    Ex: drivers, firmware, hypervisor, kernel, módulos, power.

17. **/tmp:** Arquivos temporários pelas aplicações etc. Quando reinicia, ele apaga.

    Ex: sessões do usuário, impressão temporária, downloads, arquivos de troca (swap), compressão e descompressão, cache de aplicativos.

18. **/usr:** Maior diretório do Linux depois do raiz, tem os libs, sbin, bin, mas que não são essenciais para o sistema.

    Ex: games, include, libs, share (documentação, comp dados etc.), src (código fonte dos programas), local (a 3ª hierarquia do Linux).

19. **/var:** Armazena arquivos que são gravados frequentemente pelos programas do sistema, como logs, emails, cache, etc.

    Ex: /var/log/syslog (log do sistema).

20. **/srv:** Dados que não são dinâmicos → pouco utilizado.

    Ex: páginas do http apache, serviços ftp, repositórios git.

## Desligar

- `halt`: Desligar.
- `sudo init 0`: Desliga a máquina.
- `poweroff`: Desliga.
- `shutdown -h now:` Desligar agora.
- `reboot`: Reiniciar.
- `init 6`: Reinicia.
- `shutdown -h 13:00.`
- `shutdown -h +30` "O sistema será desligado para manutenção às 23:00. Por favor, salve seu trabalho."
- `shutdown -c` “Já resolvi. Não sou reiniciar.”

▼ ***Mudando Hora | WHO e ID***

## WHO e ID

```
who -H #mostra um cabeçalho
who -a #informações detalhadas

id ubuntu #mostra o identificador
```

# MUDANDO HORA E DIA

```
sudo date -s "2024-08-02 00:15:00"
```

▼ ***History | Caminho Absoluto e relativo | Mudando Hora***

```
'echo': O comando echo é usado para exibir texto na tela.

#exemplo: echo "Olá, mundo!"
```

```
`history`: O comando histor
y mostra o histórico de comandos que você digitou anterior

#exemplo: `history | grep -E "^ *[0-9]+  (ls|cd|mkdir)"`
         #`history | grep "$(date --date='3 days ago' '+%Y
```

```
`history !3` → Pegar um comando já digitado em específico
```

```
history 3 → apenas os 3 últimos
```

```
histsize=500 →  declarar tamanho para o histórico
```

```
!-3 → 3 comando de baixo pra cima
```

```
!! → último comando
```

```
cal 5 2030 → calendário
```

```
ctrl + r → para pesquisar caso não queira pelo history
```

```
exit ou logout: O comando exit (ou logout) é usado para sa
```

```
source ou .: O comando source (ou .) é usado para executar

ex: source meu_script.sh
```

```
Use o comando bg para retomar a execução do processo em se
```

```
HISTTIMEFORMAT='%F %T' history

#se voce quiser que permanentemente, coloque export HISTTI
```

1. **Caminho Absoluto:**
   - Um caminho absoluto especifica o local de um arquivo ou diretório em relação à raiz do sistema de arquivos.
   - Começa com uma barra (`/`) e inclui todos os diretórios necessários para alcançar o arquivo ou diretório desejado a partir da raiz.
   - Exemplos de caminhos absolutos:
     - `/home/usuario/arquivo.txt`
     - `/var/log/syslog`
     - `/usr/bin/gcc`
2. **Caminho Relativo:**

- Um caminho relativo especifica o local de um arquivo ou diretório em relação ao diretório atual (ou de trabalho) em que você está.
- O diretório atual pode ser representado então por um ponto e uma barra em sistemas windows `(.\)` e inclui um diretório ou subdiretórios.

### *Exemplo*

1. Para você entender a hierarquia dos diretórios, vamos apresentar o caminho absoluto do arquivo, que é `C:\Users\Camila\Documents\Documento.tx`;
2. Eu estou no diretório `C:\Users\Camila>` e quero acessar o arquivo `Documento.txt`. Como faço isso com caminho relativo?
3. Nessas condições, o nome do caminho relativo para o arquivo ficará `.\Documents\Documento.txt`.

# MUDANDO HORA E DIA

```
sudo date -s "2024-08-02 00:15:00"
```

▼ ***Links***

## Links (Atalhos)

- **Link Simbólico (Soft Link)**:

```
ln -s /home/usuario/arquivo /home/usuario/atalho
```

- **Link Físico (Hard Link)**:

```
ln /home/usuario/arquivo /home/usuario/atalho
```

▼ ***Diretórios e ls***

## Manipulação de Diretórios

- **Criar Diretórios em Cascata**:

```
1. mkdir -p dir0/dir{1,2,3}


2. mkdir [opções] [caminho/diretório] [caminho1/diretóri
o1]

3. mkdir /diretorio{subdiretorio1, subdiretorio2, subdir
etorio3}
```

- **Contar Tamanho do Diretório**:

```
bash
Copiar código
du -sh /home/samuel
```

# MANIPULAÇÃO DE DIRETÓRIOS I

1. `-l`: exibe uma lista longa que fornece informações detalhadas sobre os arquivos, incluindo permissões, proprietário, grupo, tamanho e data de modificação.
2. `-L` : Exibe link simbólico
3. `-a` Mostra arquivos ocultos

`-A` → Lista todos os arquivos e diretórios, exceto os diretórios especiais "." (atual) e ".." (pai) e seus arquivos.

4. `ls -b` → **ver caracteres especiais em nomes de arquivo.**
5. `h, --human-readable:` Faz com que o ls mostre tamanhos de arquivo legíveis para humanos, como "1K", "234M" ou "2G".
6. `i, --inode:` Exibe o número do índice (inode) de cada arquivo. Os inodes são números únicos associados a cada arquivo no sistema de arquivos.
7. `t:` Classifica os arquivos por data de modificação, mostrando os mais recentes primeiro.
8. `r, --reverse:` Inverte a ordem de classificação, mostrando os arquivos do mais antigo para o mais novo.
9. `R, --recursive:` Lista subdiretórios de forma recursiva, ou seja, exibe o conteúdo de subdiretórios também.
10. `F, --classify:` Adiciona indicadores aos nomes dos arquivos para indicar o tipo de arquivo (por exemplo, "/" para diretórios).
11. `-color[=WHEN]:` Ativa a colorização da saída do ls para melhorar a legibilidade. "WHEN" pode ser "always" (sempre), "auto" (automático) ou "never" (nunca).
12. `G, --no-group:` Não exibe o nome do grupo dos arquivos na lista longa.
13. `d, --directory:` Lista apenas os diretórios, não seu conteúdo.
14. `s, --size:` uso de espaço em disco de cada arquivo, mostrando o número de blocos ocupados.
15. `S, --size:` Exibe do maior para o menor.
16. `q, --hide-control-chars:` Substitui caracteres não gráficos por "?" na saída.
17. `L, --dereference:` Mostra informações do arquivo referenciado por links simbólicos, em vez do link em si.
18. `c`: Classifica os arquivos com base na data de mudança (ctime), em vez da data de modificação.
19. `G, --no-group:` Não exibe o nome do grupo dos arquivos na lista longa.
20. `-n` exibe seus IDs numéricos (UID e GID).

21. `-X` listagem de arquivos por extensão, em ordem alfabética

- `ls /home/joao como ls` ~ para listar os arquivos de seu diretório home.
- `ls ../../var/log`
- `ls -lS` maior para o menor
- `ls -lrS` menor para o maior (mais antigo)
- `ls -lt` /etc  data de criação
- `ls -R /etc` lista recursivamente
- `ls -lha` → arquivos e diretórios com detalhado
- `ls -lh` → lista com os KB
- `ls -a` → todos os arquivos ocultos e diretórios especiais
- `ls -A` → Lista todos os arquivos e diretórios, exceto os diretórios especiais "." (atual) e ".." (pai) e seus arquivos.
- `ls -lha | more (ou less)`
- `ls -b` → **ver caracteres especiais em nomes de arquivo.**
- `cd /../../`
- `ls -l` → **Informações detalhadas dos diretórios etc.**
- `ls -l -f (ou -p)` → Informações com ordem de criação
- `ls -l -G (ou -o)` → Vai exibir só a coluna do arquivo
- `ls -ln` → converte os nomes do proprietário e do grupo para seus respectivos IDs de usuário (UID) e IDs de grupo (GID) numéricos
- `ls -l -L` → exibirá informações sobre o destino real de um link simbólico, em vez de exibir informações sobre o próprio link.
- `ls -t` → ordena os arquivos por data recente
- `ls -latr` → ordena os arquivos por data antiga
- `ls -lac` → ordem de criação
- `ls -laX (-lX, -lXr)` → arquivos em ordem alfabética por extensão. Isso é útil quando você deseja agrupar arquivos com a mesma extensão

juntos na lista.

- `ls -laR` → lista os arquivos e diretórios em todos os níveis da hierarquia de diretórios.

▼ ***cat | rmdir | cp***

## Remover Diretórios

- `rmdir` → remove DIRETÓRIOS VAZIOS
- `rm ~/nada/diretorio01/pokemon.txt` → *remove o arquivo "pokemon.txt"*
- `rmdir ~/nada/diretorio01 ~/nada/diretorio02/diretorio02_2 ~/nada/diretorio02` → remove os diretorios e subdiretorios
- `tree -A` → ver a árvore de diretórios criada

**removendo arquivos e diretórios**

- `rm` → remove arquivos ou diretórios
- `rm -r Diretorio1` → melhor do que o rmdir, porque ele solicita a confirmação para cada subdiretório
- `rm -rf Diretorio17/` → remover todo o diretorio (subdiretorio etc.)
- `rm -i` → o sistema solicitará uma confirmação antes de efetivamente excluir o arquivo
- `rm -rf *` → remove tudo
- `rm -rf a*` → todos que começam com a letra 'a' são removidos
- `rm -- -` → remove arquivos que tem o '-'

## visualizar conteúdo

- `cat` → visualizar o conteudo de um arquivo
- `cat -n teste` → número da linha, inclusive as linhas em branco
- `cat -s teste` → ocultar linhas em branco repetidas
- `cat -b teste` → enumera apenas as linhas que contem conteudo
- `cat -E teste` → adiciotrna um '$' nos espaços
- `cat -T teste` → converte o tab (espaço) em '^'
- `zcat teste.gz` → para ver o arquivo sem precisar descompactar

- `bzcat teste` → visualizar com extensão bz
- `tac teste` → imprime as linhas de um arquivo da última para a primeira

## copiar

- `cp [origem] [destino]` → copia de um lugar pro outro

- `cp ~/exercicio1/teste2/arquivo1.txt ~/exercicio1/teste1/arquivo2.txt` → move para outro lugar e mudando o nome do arquivo
- `cp -a ~/origem ~/destino` → fazendo uma **cópia completa**, incluindo subdiretórios e mantendo todas as informações dos arquivos e diretórios originais.
- `cp -r *Pasta Pasta2` → **copia todos os arquivos** da Pasta para Pasta 2
- `cp -p ~/origem ~/destino` → **preservação de arquivo de informações** como a data de modificação, as permissões de arquivo e o dono do arquivo.o arquivo "copia_do_arquivo.txt" terá as mesmas permissões e timestamps (data de modificação e acesso) que o arquivo original.
- `cp -v ~/origem ~/destino` → exibirá na tela informações sobre os arquivos sendo copiados
- `cp -u` **~/origem ~/destino** → a cópia será criada apenas se o arquivo de origem for mais recente do que a cópia existente no destino.
- `cp -uv`
- `cp documentos/* backup` → copia todos os arquivos de documentos para backup
- `cp */tmp` → copia todos os arquivos do diretorio atual para o /tmp
- `cp -vrx * diretorio/` → não copiar arquivos dentro de pastas que já estão em outros sistemas de arquivos
- `cp -rp diretorio_original/ copia_do_diretorio/` → o diretório "copia_do_diretorio" conterá todos os arquivos e subdiretórios do diretório original, e ***todas as permissões e timestamps serão mantidas.***

▼ ***mv | rename | visualizar conteudo***

## mover

- Apaga o arquivo de origem , e ai ele move
  - `mv` → move da origem para o destino, mas a origem é apagada
  - `touch nome_do_arquivo && mv nome_do_arquivo /caminho/para/a/pasta/`
  - `mv -p arquivo.txt /caminho/do/novo/diretorio`
  - `mv -f arquivo.txt /caminho/do/novo/diretorio/`
  - `mv -r diretorio_origem/ /caminho/do/novo/diretorio/`

### Renomear em Lote (Batch Rename):

`rename 's/padrao/novo_nome/' arquivos`

- `rename 's/.txt$/.bak/' *.txt`

## Visualizar conteúdo de Arquivo:

- Para visualizar o conteúdo de um arquivo, use o comando `cat`

`less`

- (/nome_que_quero_procurar): faz uma busca no texto
- `n` : para avançar (next)
- `shift + n` (para voltar no text)

`more` `(b para voltar e f para avançar)`

`head & tail`

- para ver o início ou final do texto
  - tail -n : para definir o numero de linhas

▼ ***Compactação de Arquivos***

## Arquivamento e Compactação

**Arquivamento:** Agrupar vários arquivos e/ou diretórios em um único arquivo.Tar = tape archive (arquivo de fita)

- **Compactação com `tar` e `gzip`:**
  - **Comandos `grep` Avançados:**

```
grep -E 'R$' alunos.txt         # Linhas que termi
nam com "R"
grep -E '^5' alunos.txt         # Linhas que começ
am com "5"
grep -E -v '^5' alunos.txt      # Linhas que não c
omeçam com "5"
grep -E '\bA' alunos.txt        # Linhas que começ
am com "A"
grep -E 'A\b' alunos.txt        # Linhas que termi
nam com "A"
grep -E '^[0-9]*.[ABCD]' alunos.txt
grep -E '[[:upper:]][[:space:]]+[[:lower:]]+[0-9].
*\@'
```

```
 zip [opcao] [nome_diretorio.zip] [nome do direotorio]

 zip -r diretorio01ediretorio02.zip diretorio01/ diretorio
```

```
tar [opções] [nome_do_arquivo_de_saida.tar.gz] [arquivo_ou
```

```
tar -czf arquivo.tar.gz /caminho/para/diretório
```

1. **gzip**:

gzip [opções] [arquivos]

- Utiliza o algoritmo de compressão Lempel-Ziv.
- Rápido e eficiente para compressão/descompressão.
- Menor taxa de compressão em comparação com `bzip2` e `xz`, mas é mais rápido.
- Extensão padrão dos arquivos compactados é `.gz`.
- Comumente utilizado para compactar arquivos individuais ou durante a transferência de dados na internet.
- Não suporta compactação de múltiplos arquivos/diretórios sem primeiro agrupá-los em um arquivo de arquivamento (`tar`).

1. **bzip2**:
   - Utiliza o algoritmo de compressão Burrows-Wheeler e Huffman.
   - Oferece uma taxa de compressão maior em comparação com `gzip`, mas é mais lento.
   - Extensão padrão dos arquivos compactados é `.bz2`.
   - É mais eficiente na compactação de arquivos de texto ou dados repetitivos.
   - Suporta compactação de múltiplos arquivos/diretórios sem a necessidade de agrupá-los em um arquivo de arquivamento.
2. **xz**:
   - Utiliza o algoritmo de compressão LZMA (Lempel-Ziv-Markov chain Algorithm).
   - Oferece uma das melhores taxas de compressão, mas é mais lento.
   - Extensão padrão dos arquivos compactados é `.xz`.
   - É comumente utilizado para compactar arquivos grandes ou distribuição de software.
   - Fornece uma excelente taxa de compressão para dados complexos ou arquivos binários.
3. **zip**:
   - Utiliza o algoritmo de compressão DEFLATE.

- É um formato de arquivo mais universalmente reconhecido e amplamente utilizado em sistemas Windows.
- Extensão padrão dos arquivos compactados é `.zip`.
- Suporta compactação de múltiplos arquivos/diretórios sem a necessidade de agrupá-los em um arquivo de arquivamento.
- Pode ser utilizado para criar arquivos de arquivamento que incluem estrutura de diretórios.

## Usando o tar:

- Usa-se tar.gz ou tar.gz2
- tar [opções] [arquivo/diretorio.tar.gz] [arquivo/diretorio]

```
tar -cvf arquivo.tar arquivo1 arquivo2 diretório1

    tar -A: Concatena arquivos de backup existentes em um 
    tar -c, --create: empacotar arquivos e diretórios em u
    tar -d, --diff, --compare: Compara um arquivo de backu
    tar -t, --list: Lista o conteúdo de um arquivo de back
    tar -r, --append: Anexa arquivos a um arquivo de backu
    tar -u, --update: Atualiza um arquivo de backup existe
    tar -x, --extract, --get: Extrai o conteudo do arquivo


    - t (list os comandos) : tar -tzf ~/backup/arquivo01_1

-c: Cria um novo arquivo tar.
-z: Usa o gzip para compactar o arquivo.
-f: diz ao tar onde gravar o arquivo tar resultante ou de 
-C: muda o diretorio antes de arquivar o conteudo

-t: Lista o conteúdo do arquivo tar.
-f: Especifica o nome do arquivo tar.
gzip -r --recursive = Compacta arquivos em todos os diretó

 criar um arquivo de arquivamento e compactá-lo com gzip d
```

```
tar -czf ~/backup/diretorio01.tar.gz ~/diretorio01 && mv ~

Utilizando o comando tar, liste os arquivos compactados e,
arquivamento e a compactação feita na questão anterior.
tar -tf ~/backup/diretorio01.tar.gz
tar -xzf ~/backup/diretorio01.tar.gz

        tar -cjf = gzip2


    tar -cvf arquivo.tar arquivo1 arquivo2: Cria um arquiv
    tar -xvf arquivo.tar: Extrai o conteúdo do arquivo de
    tar -tvf arquivo.tar: Lista o conteúdo do arquivo de a
```

```
Compactar: gzip arquivo.txt
Descompactar: gzip -d arquivo.gz
Exibir conteúdo: zcat arquivo.gz

Compactar: bzip2 arquivo.txt
Descompactar: bzip2 -d arquivo.bz2
Exibir conteúdo: bzcat arquivo.bz2
bzip2recover: Recupera dados de um arquivo compactado co
m bzip2 danificado.

Compactar: xz arquivo.txt
Descompactar: xz -d arquivo.xz

Compactar: zip arquivo.zip arquivo.txt
Descompactar: zip -d arquivo.zip
Excluir: zip -x arquivo.zip
Recursivo: zip -r arquivo.zip


num - ajusta a taxa d compactação (gzip -9 texto.txt)

gzip -9 *.txt = Mantendo o
arquivo original e no nível máximo de compressão
```

gzip -a = Força o gzip a criar um arquivo no formato ASC II, ignorando bytes nulos.

gzip -c = deseja compactar um arquivo e ver a saída comp actada sem substituir o arquivo original

gzip -v = Modo verbose, exibe mensagens detalhadas de co mpactação/descompactação.ARQUIVOS QUE ESTAO SENDO COMPAC TADOS

gzip -d =  Descompacta os arquivos compactados

gzip -f = Força a compactação

gzip -k = Mantém o arquivo original após a compactação (não o remove).

gzip -l --list = Lista informações sobre os arquivos com pactados sem descompactá-los.

gzip -L --license = Exibe a licença do software gzip .
gzip -n --no-name = Inclui o nome do arquivo original no cabeçalho do arquivo compactado.

gzip -N --name = Não inclui o nome do arquivo original n o cabeçalho do arquivo compactado.

gzip q --quiet =  Suprime a maioria das mensagens de avi so e erro durante a execução do gzip.

gzip -r --recursive = Compacta arquivos em todos os dire tórios e subdiretórios de forma recursiva.

gzip -s = Reduz a memória usada pelo gzip durante a comp actação, mas pode aumentar o tempo de compactação.Bom co

```
m máquinas com poucos recursos
```

- Usado para o gzip, bzip2, xz, zip:
  - zip [opções] [arquivo-destino] [arquivos-origem]
  -

zip -r projeto.zip . -x '*.mp4'

```
bash
Copiar código
tar -cvf site.tar /teste          # Criação do tar
gzip site.tar                      # Compactação com gzi
p
tar -czvf site.tar.gz /teste       # Compactação e arqui
vamento com gzip
tar -cjvf site.tar.bz2 /teste      # Compactação e arqui
vamento com bzip2
tar -xjvf site.tar.bz2             # Extração do arquivo
bzip2
```

▼ ***Expressões Regulares***

# Expressões Regulares

```
grep [comando] ‘[expressão]’ [lugar]
```

- ***Ex:*** `grep '[^aeiou]' exemplo.txt`
- ***Ex:*** `grep -E "(governa|verea)dora?(es | s)?" arquivo.txt`

1. `^`: Início de uma string
    - `grep '^cat' arquivo.txt` encontra linhas que começam com "cat" em "arquivo.txt".
2. `\1` : ele pega letra repetida
    - `grep -E '([a-z])\1' emailsordenados.txt`
    - `grep -E '([0-9])\.\1' endereços.txt`
3. `$`: Fim de uma string
    - `grep 'dog$' arquivo.txt` encontra linhas que terminam com "dog" em "arquivo.txt".
4. `.`: Qualquer caractere (exceto `\n` nova linha)
    - `grep 'c.t' arquivo.txt` encontra "cat", "cot", "cut" em "arquivo.txt".
5. `[ ]`: Conjunto explícito de caracteres para correspondência
    - `grep '[aeiou]' arquivo.txt` encontra qualquer linha com uma vogal em "arquivo.txt".

    grep -v '[0-9]' arquivo.txt
6. `*`: 0 ou mais da expressão anterior
    - **Uso**: `grep 'ba*' arquivo.txt` encontra "b", "ba", "baa", etc., em "arquivo.txt".
7. `[^ ]`: Nenhum dos caracteres definidos
    - `grep '[^aeiou]' arquivo.txt` encontra caracteres que não são vogais em "arquivo.txt".
8. `\`: caractere especial.
    - `grep '\.' arquivo.txt` encontra linhas com o caractere '.' em "arquivo.txt".
9. `' '`: busca algo específico.
    - `grep 'r..f'` encontra palavra que comece com r e termina com f
9. **Asterisco** `(*)`**:**
    - O asterisco corresponde a zero ou mais caracteres em um nome de arquivo ou diretório.

- Exemplo: Para listar todos os arquivos de texto em um diretório, você pode usar *.txt.

10. **Ponto de interrogação `(?)`:**
    - Unico caractere em um nome de arquivo ou diretório.
    - Exemplo: Para listar arquivos que tenham um único caractere no nome seguido por ".txt", você pode usar ?.txt. ou ls a?t*
11. **Colchetes `([])`:**
    - Especificado em uma posição.**Um caractere que comece de m [ a-z ] e vá até de 'a' a 'z'**
    - **Achar lista de opção todas as palavras que começam com 'x' mas não terminam com 'a, b, c,'**
    - Exemplo: Para listar arquivos que terminem com "1" ou "2", você pode usar *[12].
    - `ls m[^abc]` → Começam com a letra `m`.Seguidos por exatamente um caractere que não seja `a`, `b`, ou `c`.
12. **Chaves `({})` - Expansão de intervalo:**
    - lista de sequência de caracteres.
    - Exemplo: Para listar arquivos que correspondam a "arquivo1," "arquivo2," ou "arquivo3," você pode usar **arquivo{1,2,3}**
    - `ls x{zd,ze}*`
        - Começam com a letra `x`.
        - Seguidos por `zd` ou `ze`.
        - Podem ter quaisquer caracteres adicionais após `zd` ou `ze`.
13. **Barra invertida ():**
    - A barra invertida é usada para escapar caracteres curinga, fazendo com que eles sejam tratados literalmente.
    - Exemplo: Se você quiser corresponder ao caractere de asterisco literalmente, use *.

**COMANDOS**

- `i`: Ignorar diferenças entre maiúsculas e minúsculas.

    - **Exemplo:** `grep -i 'padrão' arquivo.txt` encontra "padrão", "Padrão", "PADRÃO", etc.
- `v`: Inverter a correspondência (mostrar linhas que **não** contêm o padrão).
    - **Exemplo:** `grep -v 'padrão' arquivo.txt` exibe todas as linhas que não contêm "padrão".
- `r` **ou** `R`: Buscar recursivamente em diretórios.
    - **Exemplo:** `grep -r 'padrão' /diretorio` busca "padrão" em todos os arquivos dentro de `/diretorio`.
- `l`: Listar apenas os nomes dos arquivos que contêm o padrão.
    - **Exemplo:** `grep -l 'padrão' *.txt` mostra os arquivos `.txt` que contêm "padrão".
- `n`: Mostrar o número da linha onde o padrão ocorre.
    - **Exemplo:** `grep -n 'padrão' arquivo.txt` exibe "padrão" e o número da linha em que aparece.
- `c`: Contar o número de linhas que contêm o padrão.
    - **Exemplo:** `grep -c 'padrão' arquivo.txt` exibe o número de linhas que contêm "padrão".
- `w`: Correspondência de palavras inteiras (só encontra o padrão se estiver separado por espaços).
    - **Exemplo:** `grep -w 'padrão' arquivo.txt` encontra "padrão" como uma palavra inteira — Ele só vai pegar cu, se tiver cuelho ele nao pega
- `x`: Correspondência de linha inteira (só encontra se a linha inteira corresponder ao padrão).
    - **Exemplo:** `grep -x 'linha' arquivo.txt` encontra linhas que são exatamente "linha".
- `E`: Usar várias expressões regulares.
    - **Exemplo:** `grep -E 'padrão1' -E 'padrão2' arquivo.txt` encontra linhas que contêm "padrão1" ou "padrão2".
- `A [num]`: Mostrar [num] linhas após a linha correspondente.

    - **Exemplo**: `grep -A 3 'padrão' arquivo.txt` mostra 3 linhas após cada linha que contém "padrão".
- `B [num]`: Mostrar [num] linhas antes da linha correspondente.
    - **Exemplo**: `grep -B 2 'padrão' arquivo.txt` mostra 2 linhas antes de cada linha que contém "padrão".
- `C [num]`: Mostrar [num] linhas antes e depois da linha correspondente.
    - **Exemplo**: `grep -C 4 'padrão' arquivo.txt` mostra 4 linhas antes e 4 linhas depois de cada linha que contém "padrão".

## Expressões Regulares e Busca Avançada

- **Expressões Básicas**:
    - **Alternância**: `casa | Casa`
    - **Agrupamento**: `(c|C)asas?`
- **Modificadores de Linha**:
    - `^` : Início da linha
    - `$` : Fim da linha
    - `\b` : Limite de palavra
    - `\B` : Não é limite de palavra
    - `?, * , +` : zero ou uma ocorrência

▼ ***Contagem | Corte | Comparação de palavras | Enumeração***

# WC

- contar linhas, palavras e caracteres em um arquivo ou na entrada padrão.

`wc`: Abreviação de "word count", é uma ferramenta de utilidade para contar:

- Linhas (`l`)
- Palavras (`w`)

- Caracteres (`m` ou `c`)
- Bytes (`c`)

# CUT

`cut [opções] [arquivo]`

- `f`: Especifica os campos a serem extraídos. Você deve usar em conjunto com a opção `d` para definir o delimitador.
  - Exemplo: `cut -f1,3 arquivo.txt` extrai o 1º e 3º campos do arquivo.
- `d`: Define o delimitador de campo (o caractere que separa os campos).
  - Exemplo: `cut -d',' -f2 arquivo.csv` usa a vírgula como delimitador e extrai o 2º campo.
- `c`: Especifica as posições de caracteres a serem extraídos.
  - Exemplo: `cut -c1-5 arquivo.txt` extrai os caracteres da posição 1 a 5 de cada linha.
- `s`: Supressão de linhas que não contêm o delimitador.
  - Exemplo: `cut -d',' -f1 -s arquivo.txt` só mostra linhas que contêm uma vírgula.

# *CORTANDO CAMPOS*

`cut -d ":" -f 1 /etc/passwd` - pega o primeiro campo

`cut -d ":" -f 1, 2 /etc/passwd` - pega o primeiro e segundo campo

`cut -b 1-4 /etc/passwd` - pega os bytes de 1 ao 4, pega todos os espaços

`cut -c 1-4 /etc/passwd` - pega os bytes de 1 ao 4, porém sem contar os espaços

### *COMPARAR ARQUIVOS*

`cmp arquivo1 arquivo2` - compara os 2 arquivos

`diff` - é um cmp em mais legível

`diff -u , -r` - comparar as pastas, com um símbolo de '+' e '-'

# ENUMERAÇÃO DE LINHAS

`head -c 10 passwd` → **10 primeiros bytes**

`cat /etc/passwd | sort -t ":" +2 -n | head -n 1` → **10 primeiros bytes**

`nl /etc/passwd` → **enumerada, de maneira bonita, as linhas**

`nl -f a -i 2 /etc/passwd` → **vai de 2 em 2**

`ln -f a -i 3 -v 2 /etc/passwd` → **começar por 0 a enumerar**

## MORE, LESS e SORT

**exemplo:** `cat /etc/ssh/ssh_config | more`

**exemplo:** `cat /etc/ssh/ssh_config | less` → **consigo rolar para cima e baixo.Para sair pressionar o 'q' Eu também consigo procurar coisas dentro do arquivo.Basta digitar o '/'.**

`sort arquivo2.txt` → **ordena os arquivos, primeiro os números e depois os nomes.**

`cat arquivo2.txt | sort -r` → **ordena os nomes para depois os arquivos**

`cat arquivo2.txt | sort -n` → **ordena por ordem númera.Se você não colocar o '-n', ele vai ordenar como se fosse string, pois ele entende como string.**

`cat arquivo2.txt | sort -c` → **diz se tá desordenado ou não**

`cat arquivo2.txt | sort +1` → **tá ordenando pela segunda coluna.Nesse caso se eu colocar Allyson Augusto, ele ordenará o 'Augusto', em ordem alfabética**

`cat arquivo2.txt | sort +1 -t ''F''` → **Quando ele ver a letra ''F'', ele vai ver que, tudo que está antes da letra ''F'' é a coluna 0 e posterior é coluna 1**

`cat arquivo2.txt | sort -k 1` → **especificar que a primeira coluna (campo) deve ser usada como chave para a ordenação**

▼ *Filtragem de palavras*

# *FILTRANDO E BUSCANDO INFORMAÇÕES*

## *PROCURANDO ARQUIVOS E PASTAS*

- `du -hs` mostrará o tamanho total
- `find / -name "arquivo.txt"`
  - Procura por arquivos e diretórios chamados "arquivo.txt" a partir do diretório raiz `/`.
- `find . -name nome_do_arquivo`
  - Procura por arquivos e diretórios com o nome especificado a partir do diretório atual (`.`).
- `find . -type d -name mc`
  - Encontra apenas diretórios chamados "mc" a partir do diretório atual.
- `find . -type f -name mc`
  - Encontra apenas arquivos chamados "mc" a partir do diretório atual.
- `find /usr -maxdepth 2 -type f -name mc`
  - Procura por arquivos chamados "mc" dentro do diretório `/usr`, mas apenas até dois níveis de subdiretórios.
- `find . -mtime -1`
  - Lista arquivos que foram modificados no último dia.
- `find . -ctime -1`
  - Lista arquivos que foram criados no último dia.
- `find . -atime -1`
  - Lista arquivos que foram acessados no último dia.
- `find /usr -size +1000`

    - Lista arquivos dentro do diretório `/usr` que têm mais de 1000 bytes.
- `free --kilo ou --mega ou --kibi --mebi --gibi --giga` (memória em Kbytes ou Megabytes) → porque (quilo, mega, giga, etc.) usados no sistema métrico são baseados em potências de 10, enquanto as unidades de armazenamento de dados em computação são baseadas em potências de 2.
- `free --mega -s 1` → me traz a memória utilizada a cada 1s

# *PROCURANDO TEXTOS DENTRO DE ARQUIVOS*

1. `grep ‘r.d’ red.txt`: Imprimirá linhas do arquivo 'red.txt' que contenham um 'r' seguido de qualquer caractere e depois 'd'.
2. `grep 'root' /etc/passwd`: Procurará por linhas no arquivo '/etc/passwd' que contenham a palavra 'root'.
3. `grep -v 'www-data' /etc/passwd`: Imprimirá todas as linhas do arquivo '/etc/passwd' exceto aquelas que contenham 'www-data'.
4. `grep -f /tmp/`: Procurará padrões contidos nos arquivos listados em '/tmp/'.
5. `grep -i 'WWW-DaTa' /etc/passwd`: Procurará por 'WWW-DaTa' no arquivo '/etc/passwd' sem diferenciação entre maiúsculas e minúsculas.
6. `grep -iE '^www-data.*nologin$'`: Procurará no arquivo '/etc/passwd' por linhas que começam com 'www-data' e terminam com 'nologin'.
7. `grep -iF '.*' /etc/passwd`: Imprimirá linhas do arquivo '/etc/passwd' que contenham pelo menos um caractere.
8. `grep -ri 'Foca02' .`: Procurará recursivamente por 'Foca02' nos arquivos e diretórios atuais, ignorando diferenças de maiúsculas e minúsculas.
9.

`grep -rih`: Procurará recursivamente, ignorando maiúsculas e minúsculas, mas não imprimirá os nomes dos arquivos correspondentes.
10.
`grep -ril 'Foca02' .`: Listará os arquivos que contêm 'Foca02', mas não imprimirá o conteúdo dos arquivos.

- `head passwd` → **mostra por padrão as 10 primeiras linhas**
- `head -n 3 passwd` → **3 primeiras linhas**
- `cat passwd | head -n 3`
- `echo www-data > /tmp/expressao` → **cria algo e já coloca em um lugar**

▼ ***Comandos Diversos (Date, Time, Dmesk, Tail e Head, wc)***

# COMANDOS DIVERSO

***DATE***

- `sudo date --set="2024-08-02 22:30:00”`
- `sudo date 101007452020` (mes, dia, hora, minuto e ano) e para salvar na CMOS do sistema, utiliza-se o **hwclock --systohc**
- `date + "%d-%m-%Y %T"`
- `date +%R` → (formato de 24 horas)
- `date +"%d %Y %j"` → Quantos dias falta pra acabar o ano
- `df -h e df-Th` → Mostra as partições e seus respectivos sistemas de arquivos
- `df -aTh -t arquivo_especifico` → Busca uma especificação nas partições
- `ln -s minha_pasta meu_link` → Link simbólico que, qualquer operação realizada dentro de "meu_link", afetará o conteúdo de "PASTA".Ou seja, ln -s /usr/bin bin-usr. Toda vez que eu entrar no caminho /usr/bin ele vai para bin-usr

# TIME e UPTIME

`uptime` → *tempo de atividade da máquina desde o último boot*

`time ls` → *veja o tempo de execução do comando*

# DMESG, TALK e MESG

`dmesg` → *fornece informações sobre o hardware, drivers de dispositivos e eventos do kernel do sistema*

```
dmesg -t | grep enp0s3
```

`dmesg -x` → *prioridade das mensagens em texto legível*

`dmesg -T` → *legível*

`dmesg -c` → *limpa as mensagens do buffer do kernel para esperar as outras*

`mesg` →*definir se você permite ou não que outros usuários enviem mensagens de bate-papo para você.Se digitar 'mesg y' ele ativa.Para desativar, digite ‘mesg n’*

`talk` → *mandando mensagem em tempo real*

# ECHO

`echo` → *mensagem na tela*

`echo -n "teste no linux admin"` → *não faz quebra de linha*

`echo -e "Teste do Linux" -` → *habilita os caracteres especiais*

# DESLIGAR A MÁQUINA

`sync` - *gravar os buffers do kernel no disco, porque ao invés de esperar 20s , ele faz forçado*

`uname -a, -r, -n` -- *nome da máquina e suas especificações*

`echo b>/proc/sysrq-trigger` - *reiniciar para emergencia*

`halt - desligar`

`echo o>/proc/sysrq-trigger` --> *desligar a máquina forçada, caso esteja com problemas de emergência*

`shutdown -h 09:40` --> *agendamento para desligar*

`wc` - *retorna palavras, bytes e linhas de um arquivo (ex: wc /etc/passwd)*

`seq` - *sequência de números (ex: seq 10, seq 2 2 10, seq 2 10)*

## TOUCH

`touch -t 10120815 /tmp/arquivo` → ***modifica a hora e data do arquivo0***

`touch -a -t 10120815 /tmp/arquivo` → ***modifica o acesso do arquivo***

ERROS:

# TAIL , HEAD e LESS

```
head arquivos_com_a.txt   # Exibe as primeiras 10 linhas

less arquivos_com_a.txt   # Rolar para cima e para baixo n
```

```
tail arquivos_com_a.txt   # Exibe as últimas 10 linhas
tail -f /var/log/auth.log → #pega as última linhas do arqu
tail /etc/passwd --> #visualiza as 10 ultimas linhas
tail -n 4 /etc/passwd --> #visualiza as 4 ultimas linhas
tail -f /var/log/auth.log →
```

```
grep '^[[:space:]]*A' palavras.txt
```

```
grep -E "[0-9]{2}/[0-9]{2}/[0-9]{4}" datas.txt
```

```
grep "[?$]" perguntas.txt
```

```
grep -E '[a-z]{6}' dicionario.txt
```

```
grep -E '([0-9]{1,3}\.){3}[0-9]{1,3}' logs.txt
```

```
grep -E 'http[s]?://[a-zA-Z0-9./?=]+' links.txt
```

```
grep -E '[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,7}'
```

## WC

- `wc -l` #Linhas
- `wc -w` #Palavras
- `wc -c` # Caracteres

## HEAD

- `head -n teste01.txt` #exibir as linhas

## TAIL

```
tail arquivos_com_a.txt → Exibe as últimas 10 linhas
tail -f /var/log/auth.log → pega as última linhas do arqui
tail /etc/passwd → visualiza as 10 ultimas linhas
tail -n 4 /etc/passwd → #visualiza as 4 ultimas linhas
tail +numero →
```

▼ ***Redirecionamentos e Pipes | Filtrando e buscando informações***

# REDIRECIONAMENTO E PIPES

- Todo shell em um sistema operacional precisa comunicar-se com o usuário por meio de dispositivos de entrada e saída.
- Um shell Linux, como bash, recebe entrada e envia saída como sequências ou fluxos de caracteres.
  - Cada caractere é independente do que vem antes e do que vem depois dele.
  - Os caracteres não são organizados em registros estruturados ou blocos de tamanho fixo.
- Fluxos são acessados utilizando técnicas de E/S (**E**ntrada/**S**aída) de arquivo,
  - Não importa se o fluxo de caracteres real vem de ou vai para um arquivo, um teclado, uma janela em um monitor ou outro dispositivo de E/S.

- Shells Linux usam três fluxos de E/S padrão, cada um dos quais é associado com um descritor de arquivo:
  - **Entrada padrão (stdin) –** Entrada de um fluxo de dados, podendo ser destacado o teclado.
    - Fluxos de entrada fornecem entrada para programas, normalmente de digitações em um terminal.
    - O descritor é representado pelo **número 0**.
  - **Saída padrão (stdout) –** Saída de um fluxo de dados em condições normais. Exemplos: monitor, impressora, arquivos, etc.
    - Fluxos de saída imprimem caracteres de texto.
    - O descritor é representado pelo **número 1**.
  - **Saída de erro (stderr) –** Saída de um fluxo de dados em condições de erro ou insucesso em um determinado processamento, que poderá ser direcionada para o monitor ou arquivo de LOG.
    - O descritor é representado pelo **número 2**.

**>**

- Redireciona a saída padrão de um programa/comando/script para algum arquivo ao invés do dispositivo de saída padrão (tela).Ele sobrescreve

```
Exemplo:
– ls > teste.txt : Envia a saída do comando ls para o arqu
teste.txt.
– ls > /dev/tty2 : Envia a saída do comando ls para o segu
console.
```

**>>**

- Adiciona as linhas ao final do arquivo ao invés do dispositivo de saída padrão (tela).Basicamente ele dá um 'append' e ele não sobrescreve
- `Ex: ls /usr/fake > deucerto.txt2> deuerrado.txt`:
- se o diretório `/usr/fake` existir, a lista de arquivos será salva em `deucerto.txt`, e se não existir, a mensagem de erro será salva em `deuerrado.txt`.

```
Exemplo:
– ls >> teste.txt : Adiciona a saída do comando ls ao fina
arquivo teste.txt, se ele existir.
– ls >> /dev/tty2 : Envia a saída do comando ls para o
segundo console

echo "testando 1234..." >> arquivo1.txt
```

**<**

- Faz com que o comando leia os dados do arquivo especificado em vez de esperar por entrada do teclado.

```
cat < teste.txt

Isso faz com que o conteúdo do arquivo teste.txt seja envi
```

**<<**

- insira várias linhas de entrada interativamente, terminando a entrada quando uma determinada palavra-chave é digitada.

```
#!/bin/bash

# Este script solicita ao usuário para inserir seu nome
# e, em seguida, imprime uma saudação personalizada.

echo "Por favor, insira seu nome:"
# O operador '<<' redireciona a entrada para o comando 're
# a partir de uma sequência de caracteres terminada pela p
read -r nome << EOF
EOF

echo "Olá, $nome! Bem-vindo!"
```

**tr**

- Troca/substituição de caracteres
- Não aceita arquivo como argumento

```
tr 'a-z' 'A-Z'

echo "Hello World" | tr '[:lower:]' '[:upper:]'
```

# CONECTORES

## | (Pipe)

- **Descrição**: Conecta a saída de um comando como entrada para outro comando.
- **Exemplo**:

```
cat a.txt b.txt | grep nome
```

## ; (Ponto e Vírgula)

- **Descrição**: Usado para executar vários comandos em sequência, independentemente do sucesso ou falha de cada um.
- **Exemplo**:

```
echo "Arquivo 1"; cat $1; echo "Arquivo 2"; cat $2
```

## && (E Lógico)

- **Descrição**: O segundo comando só é executado se o primeiro for bem-sucedido (retornar código 0).
- **Exemplo**:

```
mkdir teste && echo "Diretório criado com sucesso"
```

### `||` (Ou Lógico)

- **Descrição**: O segundo comando só é executado se o primeiro falhar (retornar código diferente de 0).
- **Exemplo**:

```
mkdir teste || echo 'Não foi possível criar o diretór
io "teste"'
```

## tee

- ler a entrada padrão e escrever tanto na saída padrão quanto em um ou mais arquivos
    - Por exemplo, se você deseja visualizar o conteúdo de um arquivo e, ao mesmo tempo, salvá-lo em outro arquivo, você pode usar o `tee`

```
cat arquivo.txt | tee novo_arquivo.txt
```

# FILTRANDO E BUSCANDO INFORMAÇÕES

## sort

- classificar as linhas de texto em ordem alfabética ou numérica

- `t`: Define o delimitador para separar os campos de cada linha.
- `k3`: Especifica o campo a ser usado como chave para ordenação.

- `n`: Indica que a ordenação deve ser numérica.
- `r`: Reverte a ordem de classificação, tornando-a decrescente.

```
cat arquivo.txt | tee novo_arquivo.txt
```

```
Ordenar um arquivo de texto em ordem alfabética:

sort arquivo.txt


Ordenar um arquivo de texto em ordem numérica:

sort -n numeros.txt


Ordenar um arquivo de texto em ordem inversa:

sort -r arquivo.txt


Ordenar um arquivo de texto ignorando maiúsculas e minúscu

sort -f arquivo.txt


Ordenar um arquivo de texto e remover linhas duplicadas:
```

```
sort -u arquivo.txt



Ordenar um arquivo de texto e salvar a saída em um novo ar



sort entrada.txt -o saida.txt



Ordenar um arquivo de texto considerando apenas os primeir



sort -k1.1,1.3 arquivo.txt
sort -t, -k1n -k3 sort.txt









Suponha que temos o seguinte arquivo chamado "dados.txt":

1,Ana,25
2,João,30
3,Maria,22

Podemos ordenar este arquivo com base na terceira coluna (

bash

sort -t ',' -k3n dados.txt

Isso produzirá a seguinte saída:
```

```
3,Maria,22
1,Ana,25
2,João,30

Explicação do comando:

    -t ',': Define a vírgula como delimitador de campo.
    -k3n: Especifica que queremos classificar com base no
    dados.txt: O nome do arquivo de entrada.
```

**cut (cortando campo)**

`cut -d ":" -f 1 /etc/passwd` - pega o primeiro campo

`cut -d ":" -f 1, 2 /etc/passwd` - pega o primeiro e segundo campo

`cut -b 1-4 /etc/passwd` - pega os bytes de 1 ao 4, pega todos os espaços

`cut -c 1-4 /etc/passwd` - pega os bytes de 1 ao 4, porém sem contar os espaços

```
cut OPÇÕES ARQUIVO

Aqui estão algumas opções comuns do comando cut:

    -c LISTA: Especifica quais caracteres devem ser incluí
    -f CAMPOS: Especifica quais campos devem ser incluídos
    -d DELIMITADOR: Especifica o caractere delimitador de

Exemplo de uso:

Suponha que temos um arquivo chamado "nomes.txt" com o seg

João, Silva, 25
Maria, Santos, 30
```

```
Para extrair apenas o primeiro campo (nome) deste arquivo,

bash

cut -d ',' -f1 nomes.txt

Isso produzirá a seguinte saída:

João
Maria
```

# grep

## Comandos Básicos

- `grep 'root' /etc/passwd`

  Encontra todas as linhas no arquivo `/etc/passwd` que contêm "root".

- `grep -v 'www-data' /etc/passwd`

  Inverte a busca, exibindo linhas exceto as que contêm "www-data".

- `grep -f /tmp/patternfile /etc/passwd`

  Usa um arquivo (`/tmp/patternfile`) para definir o padrão de busca.

## Opções de Contexto

- `A [número]`

  Mostra o número de linhas após a linha encontrada.

- `B [número]`

  Mostra o número de linhas antes da linha encontrada.

- `C [número]`

  Mostra o número de linhas ao redor da linha encontrada (antes e depois).

## Opções de Contagem e Numeração

- `c`

  Conta quantas linhas correspondem ao padrão.

- `n`

  Exibe os números das linhas correspondentes.

## Opções de Correspondência e Sensibilidade ao Caso

- `w`

  Retorna apenas linhas que contêm correspondências como palavras inteiras.

- `i`

  Realiza uma busca case-insensitive (ignora maiúsculas e minúsculas).

- `grep -i 'WWW-DaTa' /etc/passwd`

  Busca case-insensitive por "WWW-DaTa" no arquivo `/etc/passwd`.

## Opções de Expressões Regulares e Literais

- `E`

  Usa expressões regulares estendidas.

- `grep -iE '^www-data.*nologin$' /etc/passwd`

  Encontra linhas que começam com "www-data" e terminam com "nologin" (case-insensitive).

- `F`

  Interpreta o padrão como uma string literal (não regex).

- `grep -iF '.*' /etc/passwd`

  Interpreta ".*" como caracteres literais, útil para buscas de caracteres especiais.

## Opções de Busca Recursiva

- `r`

  Realiza busca recursiva nos diretórios.

- `ri 'Foca02' .`

Busca recursivamente, ignorando o case, por "Foca02" no diretório atual.

- `rh`

  Oculta os nomes dos arquivos nos resultados da busca recursiva.

- `grep -ril 'Foca02' .`

  Lista os nomes dos arquivos que contêm "Foca02", sem mostrar o conteúdo.

## Pesquisa em Arquivos ( `grep` )

```
grep -c "palavra" #conta o número de linhas que contêm a palavra
grep -i "palavra" arquivo = comando #busca a palavra ignorando maiúsculas e minúsculas
grep -l "palavra" #lista o nome do arquivo se contiver a palavra
grep -v "palavra" arquivo #exibe as linhas que não contêm a palavra
grep -o "palavra" arquivo #mostra apenas a palavra encontrada, sem a linha inteira
grep -n soma arquivo #exibe o numero da linha
```

- `grep 'root' /etc/passwd`

  Encontra todas as linhas no arquivo `/etc/passwd` que contêm "root".

- `grep -v 'www-data' /etc/passwd`

  Inverte a busca, exibindo linhas exceto as que contêm "www-data".

- `grep -f /tmp/patternfile /etc/passwd`

  Usa um arquivo (`/tmp/patternfile`) para definir o padrão de busca.

### Opções de Contexto

- `A [número]`

  Mostra o número de linhas após a linha encontrada.

- `B [número]`

Mostra o número de linhas antes da linha encontrada.

- `C [número]`

  Mostra o número de linhas ao redor da linha encontrada (antes e depois).

## Opções de Contagem e Numeração

- `c`

  Conta quantas linhas correspondem ao padrão.

- `n`

  Exibe os números das linhas correspondentes.

## Opções de Correspondência e Sensibilidade ao Caso

- `w`

  Retorna apenas linhas que contêm correspondências como palavras inteiras.

- `i`

  Realiza uma busca case-insensitive (ignora maiúsculas e minúsculas).

- `grep -i 'WWW-DaTa' /etc/passwd`

  Busca case-insensitive por "WWW-DaTa" no arquivo `/etc/passwd`.

## Opções de Expressões Regulares e Literais

- `E`

  Usa expressões regulares estendidas.

- `grep -iE '^www-data.*nologin$' /etc/passwd`

  Encontra linhas que começam com "www-data" e terminam com "nologin" (case-insensitive).

- `F`

  Interpreta o padrão como uma string literal (não regex).

- `grep -iF '.*' /etc/passwd`

  Interpreta ".*" como caracteres literais, útil para buscas de caracteres especiais.

## Opções de Busca Recursiva

- `r`

  Realiza busca recursiva nos diretórios.

- `ri 'Foca02' .`

  Busca recursivamente, ignorando o case, por "Foca02" no diretório atual.

- `rh`

  Oculta os nomes dos arquivos nos resultados da busca recursiva.

- `grep -ril 'Foca02' .`

  Lista os nomes dos arquivos que contêm "Foca02", sem mostrar o conteúdo.

```
grep -v '^$'
```

▼ ***Permissões***

▼ CHOWN e CHGRP (1)

- Serve para mudar o grupo/dono do arquivo/diretório

```
chown giropops teste
```

```
chown giropops:users teste ou chown giropops.users test
```

```
chown :jeferson teste
```

**CHGRP**

```
chgrp jeferson teste_dir/
```

▼ STICK BIT, SUID, SGID e UMASK (1)

## STICK BIT

- representado pela letra t e na maneira octal pelo ‘1’
- apenas o proprietário do arquivo, o proprietário do diretório e o superusuário podem renomear ou excluir arquivos dentro desse diretório, mesmo que outros usuários tenham permissão de escrita no diretório.


```
chmod +t nome_do_diretorio
```

## SUID

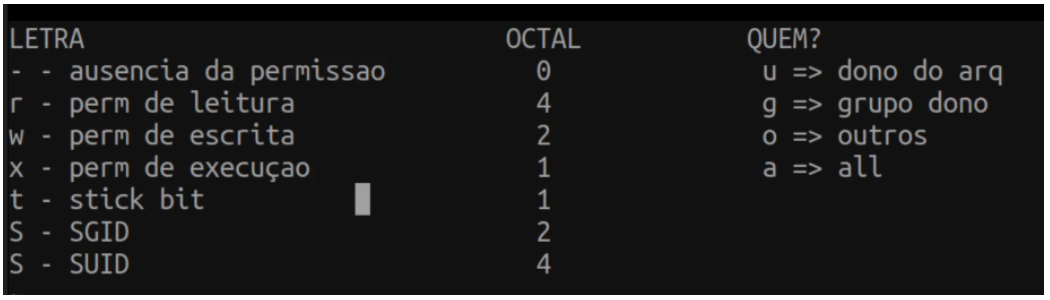
- Todo arquivo binário que tiver essa permissão habilitada, qualquer pessoa que executar esse executável vai ser como se fosse dono do executável
- Representado pela ‘’S’’ e no Octal pelo ‘’4’’

```
chmod 4755 nome_do_arquivo
```

## GUID

- Todo arquivo binário que tiver essa permissão habilitada, o grupo que executar esse executável vai ser como se fosse dono do executável
- Representado pelo ‘’S’’ e no Octal pelo ‘’2’’

```
chmod 2755 nome_do_arquivo
```




# UMASK

- A máscara de permissão (umask) determina as permissões padrão de um novo arquivo ou diretório criado por um usuário no sistema operacional Unix/Linux.
- Quando você fizer logout ele não vai ter mais o mesmo mask, então voce, se quiser colocar como padrão, precisa salvar o arquivo
- Colinha: Permissão de arquivos é sempre 666.Permissão de diretório é sempre 777.A permissão padrão de quando for criar um diretório ou arquivo é o valor (arquivo ou diretorio) - mask

```
umask valor
```

▼ ***PROTEGENDO ATRIBUTOS*** (1)

`chattr` - alterar os atributos de um arquivo (imutável, escrita etc.)

`lsattr` - listar os atributos de um arquivo

`chattr +i nome_do_arquivo` - imutável (ex: chattr +i teste)

`chattr -i nome_do_arquivo` - tornar mutável novamente

`chattr +a nome_do_arquivo` - impede a sobrescrição de dados (echo "Linha 3" > teste 3) para testar

`chattr +c` - compactado

`chattr =ai *` - atribui os atributos a todos os arquivos

`chattr +D attr` - mesmo que alguém tenha permissões para escrever nesse diretório, não será possível excluir nenhum arquivo ou subdiretório dentro dele

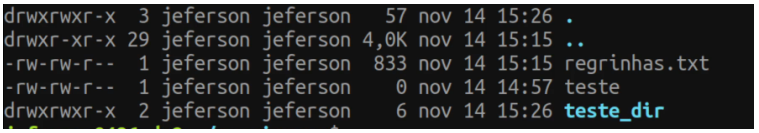
`chattr +c` - sistema operacional tenta compactar automaticamente o arquivo quando está sendo gravado no disco

`chattr +s` - realiza atualizações síncronas no arquivo

`chattr +S` - grava rapidamente no disco.Diferente do 'sync', que demora um pouco mais.

▼ chmod (1)

# PERMISSÕES




1. **Primeiro caractere: Tipo de arquivos**
   - `-`: Indica um arquivo regular.
   - `d`: Indica um diretório.
   - `l`: Indica um link simbólico.
   - `-` arquivo de texto
   - `c` Dispositivos de caractere (portas que transferem informações)
   - `s` socket
2. **Caracteres 2-4: Permissões do proprietário**
   - `r`: Permissão de leitura.
   - `w`: Permissão de escrita.
   - `x`: Permissão de execução.
   - `-`: Indica ausência de permissão.
3. **Caracteres 5-7: Permissões do grupo**
   - `r`: Permissão de leitura.
   - `w`: Permissão de escrita.
   - `x`: Permissão de execução.

- `-`: Indica ausência de permissão.

4. **Caracteres 8-10: Permissões para outros usuários**
   - `r`: Permissão de leitura.
   - `w`: Permissão de escrita.
   - `x`: Permissão de execução.
   - `-`: Indica ausência de permissão.

```
-rw-rw-r--
```

**OBSERÇÃO**: Se você tiver subpastas e quer aplicar as permissões a todas elas, basta colocar como recursivo (chmod -R 511 teste_dir/)

## COMANDOS

```
chmod - modifica as permissões
chown - modifica o dono/grupo do arquivo/diretório
chgrp - modifica o grupo do arquivo/diretório
umask - define a permissão padrão de arquivos e dire
```

exemplos - Usando maneira 'Simbólica' - CHMOD

### Permissões Básicas

- **r**: Leitura (read)
- **w**: Escrita (write)
- **x**: Execução (execute)

### Categorias de Usuários

- **u**: Proprietário (user)
- **g**: Grupo (group)

- **o**: Outros (others)
- **a**: Todos (all)

### Operadores

- **+**: Adiciona uma permissão
- **-**: Remove uma permissão
- **=**: Define permissões exatas

## Exemplos de Uso do `chmod` com Forma Simbólica

1. **Adicionar Permissão de Execução para o Proprietário**

   ```
   chmod u+x arquivo.txt
   ```

   Adiciona a permissão de execução para o proprietário do arquivo `arquivo.txt`.

2. **Remover Permissão de Escrita para o Grupo**

   ```
   chmod g-w arquivo.txt
   ```

   Remove a permissão de escrita para o grupo em `arquivo.txt`.

3. **Adicionar Permissão de Leitura para Outros**

   ```
   chmod o+r arquivo.txt
   ```

   Adiciona a permissão de leitura para outros usuários em `arquivo.txt`.

4. **Definir Permissões Exatas para o Proprietário e Grupo**

```
chmod u=rwx,g=rx arquivo.txt
```

Define permissões de leitura, escrita e execução para o proprietário e permissões de leitura e execução para o grupo em `arquivo.txt`.

5. **Remover Todas as Permissões para Outros**

```
chmod o-rwx arquivo.txt
```

Remove todas as permissões para outros usuários em `arquivo.txt`.

6. **Adicionar Permissão de Leitura para o dono do grupo**

```
chmod u=r diretorio01_1/teste.txt
```

7. restaurar as permissões originais do arquivo `teste.txt` para o dono (usuário)

```
#~/diretorio01/diretorio01_1$ ls -l

#-rw-rw-r-- 1 ubuntu ubuntu 24 Sep 13 14:33 teste.tx


chmod u=rw,g=rw,o=r diretorio01/diretorio01_1/teste.
```

## Exemplo com maneira ‘Octal’:

- Observação: NUNCA DEIXAR COMO ‘777’, pois todos vão ter todas as permissões

## Permissões Básicas e Seus Valores Octais

- **Leitura (r)**: 4
- **Escrita (w)**: 2
- **Execução (x)**: 1

As permissões são representadas por três dígitos octais:

1. **Primeiro Dígito**: Permissões para o Proprietário
2. **Segundo Dígito**: Permissões para o Grupo
3. **Terceiro Dígito**: Permissões para Outros

Cada dígito é a soma das permissões que você deseja conceder. Por exemplo:

- **7** = 4 (leitura) + 2 (escrita) + 1 (execução) = rwx
- **6** = 4 (leitura) + 2 (escrita) = rw-
- **5** = 4 (leitura) + 1 (execução) = r-x
- **4** = 4 (leitura) = r--

## Exemplos de Uso do `chmod` com Notação Octal

1. **Definir Permissões Completas para o Proprietário, Leitura e Execução para o Grupo e Nenhuma Permissão para Outros**

   ```
   chmod 750 arquivo.txt
   ```

   - Proprietário: `rwx` (7)
   - Grupo: `r-x` (5)
   - Outros: `--` (0)

2. **Definir Permissões de Leitura e Escrita para o Proprietário e Grupo, e Nenhuma Permissão para Outros**

   ```
   chmod 664 arquivo.txt
   ```

   - Proprietário: `rw-` (6)

- Grupo: `rw-` (6)
- Outros: `--` (4)

3. **Definir Permissões de Leitura e Execução para Todos**

```
chmod 555 arquivo.txt
```

   - Proprietário: `r-x` (5)
   - Grupo: `r-x` (5)
   - Outros: `r-x` (5)

4. **Definir Permissões Completas para o Proprietário e Somente Leitura para o Grupo e Outros**

```
chmod 744 arquivo.txt
```

   - Proprietário: `rwx` (7)
   - Grupo: `r--` (4)
   - Outros: `r--` (4)

5. **Remover Todas as Permissões para o Grupo e Outros**

```
chmod 700 arquivo.txt
```

   - Proprietário: `rwx` (7)
   - Grupo: `--` (0)
   - Outros: `--` (0)

## Resumo dos Comandos `chmod` Octais

- **Definir Permissões:**

```
chmod [modo octal] arquivo
```

- **Exemplos de Modos Octais:**
  - **777**: Permissões completas para todos (`rwxrwxrwx`)
  - **755**: Permissões completas para o proprietário e leitura e execução para grupo e outros (`rwxr-xr-x`)
  - **644**: Leitura e escrita para o proprietário, leitura para grupo e outros (`rw-r--r--`)

▼ ***Editor de texto***

## ESCREVER, SAIR e SALVAR

- *"`i`"* (insert) para inserir texto
- *`ESC`* -Retorna ao modo de comando (não pode inserir nada)

- *`ESC + o`* insere texto na linha abaixo
- *`ESC + O`* - insere texto na linha de cima
- *`ESC + I (i maiúsculo)`* - inserção e vai para o início da linha
- *`ESC + a`* - Inserir um caractere a frente
- *`ESC + A`* - Inserir no final da linha

```
#Entra no VI
vim giropops

#Salvando e saindo
:wq

#Salvando
:w

#Sair sem salvar
:q!

#Sair e salvar
:x
```

```
#Sai e salvar
ZZ

#Sai sem salvar
ZQ
```

## COPIAR , COLAR e APAGAR

- `yy` e `p` - copia e cola
- `SHIFT + P` - cola na linha de cima
- `cw` - recortar uma palavra
- `yw` - copiar uma palavra
- `y8y` - copiando 8 linhas

- `dd` - apagando/recortando a linha
- `dw` - apaga uma palavra
- `dG` - apaga até o final do arquivo
- `dgg` - apaga até o começo do arquivo
- `SHITF + d` - apagar do cursor até a última linha
- `d8d` - apagando/recortando 8 linhas
- `x` - remove apenas um caractere
- `X` - remove um caractere antes

## VOLTANDO e REFAZENDO

- `r + nova_letra` - substituir (replace) o caractere atual pelo novo
- `CTRL + U` - voltar
- `CTRL + Z` - refazer

## MODO VISUAL

- Se você der um `ESC` você vai para o modo “comando”, porém se você quiser ir para o modo visual, basta presionar o `v` com isso você pode selecionar uma frase etc.
- `SHIFT + v` - seleciona um bloco de texto
- `v` - seleciona um pedaço de texto
- `V` - seleciona linhas do texto

## VISUAL BLOCK

- `CTRL + v` Seleciona um bloco de texto
- ***`SHIFT + v`*** - seleciona um bloco de palavras

## BUSCAS e LOCALIZAÇÃO

- `/palavra` - pesquisar uma determinada palavra descendo o arquivo
- `?palavra` - busca a palavra subindo o arquivo
- `N` - continua a busca pra cima
- `n` - continua a busca para baixo
- `gg` - vai para a primeira linha
- `G` - vai para o final do arquivo
- `M` - meio da tela
- `H` - vai para o alto da tela
- `L` - na parte da tela

## CONFIGURANDO O VI (set)

- `:set` - para começar a setar alguma configuração em específica
- `:set number ou nu` - numera as linhas
- `:set hlsearch` - tudo que ele tiver procurando, vai ficar em negrito a palavra. Se você não quiser mais isso, basta dar um `:set nohlsearch`

- `:set tabstop=2` - tamanho do TAB
- `:set expandtab` - converte o tab em espaços
- `:set noerrorbells` - não vai ter aqueles sons de erro
- `:set novisualbell` - não ficar piscando quando tiver erro
- `:set bg=light` - muda a cor do VI
- `:set ignorecase` - ignora maiusculo de minusculo
- `:set syntax=on` - habilita a sintaxe de uma extensão (.yaml, json, python)

## SUBSTITUIR

- `:46s/palavra_antiga/palavra_nova/` - substitui a palavra de uma determinada linha (46), com o nome dela (giropops) por uma nova palavra (strigus)
- `:%s/palavra_antiga/palavra_nova/`- substitui a palavra giropops por strigus em todo o arquivo.Só substitui a primeira palavra que encontrar
- `:%s/palavra_antiga/palavra_nova/g` - substitui a palavra giropops por strigus em todo o arquivo em todas as palavras que achar
- `:40,50s/palavra_antiga/palavra_nova/` - substitui da palavra por outra da linha 40 até 50

## OUTROS

- `:e` - abre outro arquivo
- `:r BLA` copia o conteudo do arquivo BLA para o arquivo atual

  `:split example.yaml` - dividir a tela do arquivo na vertical.Se você quiser editar no de baixo, basta precionar o `CTRL + ww`

  s
- `:vsplit /etc/resolv_conf` - dividir a tela na horizontal
- `:qa` - fecha todas as janelas
- `:! ip a` - pega um IP, por exemplo, sem sair do VI.Nesse caso, coloque um `":! "`
- `!!hostname` - pega a saida e joga para dentro do VI

▼ ***Administração de usuários***

# SUPER USUÁRIO

- `Su [opções] [usuario]`
- As informações dos usuários estão dentro de `/etc/passwd` e os do grupo é em `/etc/group`
- As senhas estão dentro de `/etc/shadow` e as senhas dos grupos é no `/etc/gshadow`

# ADMINISTRAÇÃO DE USUÁRIOS

## 1. IDs de Usuários e Grupos no Sistema

- **IDs de Grupo de Sistema**: Variam conforme a distribuição, geralmente entre `0-999` ou `100-999`.
- **IDs de Usuários Normais**: Começam geralmente a partir de `1000`.

## 2. Comandos Relacionados a Usuários e Grupos

- `users`: Lista os usuários atualmente logados.
- `groups`: Exibe os grupos dos usuários.
- `id`: Mostra o ID do usuário, ID do grupo principal, e os IDs dos grupos secundários aos quais o usuário pertence.

## 3. Arquivo `/etc/passwd`

- Contém informações sobre os usuários do sistema.

```
cat /etc/passwd
```

## 4. Arquivo `/etc/group`

- O arquivo `/etc/group` armazena informações sobre os grupos no sistema

```
cat /etc/group
```

### Estrutura de uma Linha no `/etc/passwd`

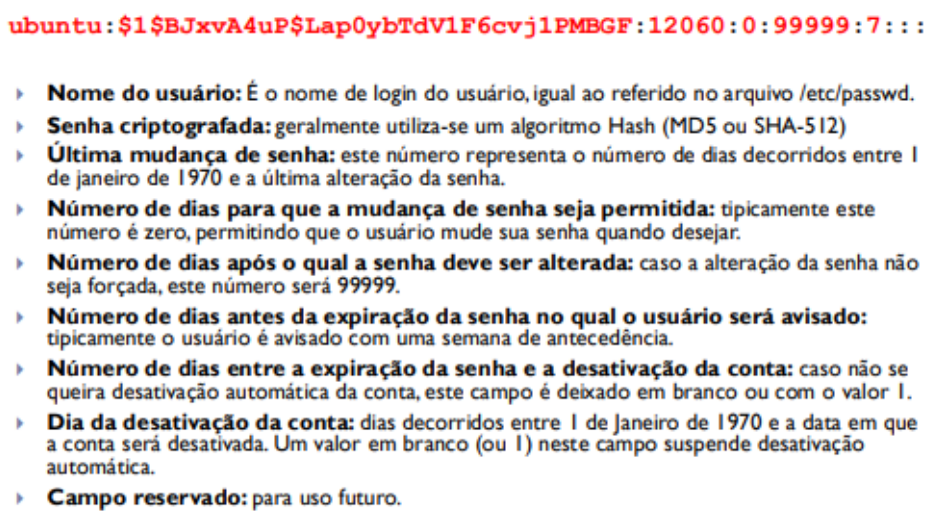
```
root:x:0:0:root:/root:/bin/bash
```

## Descrição dos Campos:

1. **Campo do Usuário**: Nome do usuário (ex.: `root`).
2. **Senha**: `x` indica que a senha está armazenada em outro arquivo (`/etc/shadow`).
3. **UID (User ID)**: `0` para o usuário root. Usuários normais começam geralmente a partir de `1000`.
4. **GID (Group ID)**: `0` para o grupo root. Corresponde ao ID do grupo principal do usuário.
5. **GECOS (Comentários)**: Informações adicionais como nome completo, número de sala, etc. (ex.: `root`).
6. **Home Directory**: Diretório inicial do usuário (ex.: `/root`).
7. **Shell**: Shell padrão do usuário (ex.: `/bin/bash`).

### Estado das Contas

- `*` **ou** `!` no campo de senha (no arquivo `/etc/shadow`):
  - `*` indica que a conta está bloqueada.
  - `!` indica que a conta está configurada sem senha.




## getent

- buscar informações da senha sobre o usuário "joao"

```
getent passwd joao
```

## Login

- usuário se autentique e acesse o sistema com suas credenciais (nome de usuário e senha).

```
login usuario
```

## 1. lastlog

- **Descrição**: Mostra o último login dos usuários cadastrados no sistema.
- **Uso**:

```
lastlog [opções]
```

- **Opções**:
  - `u [nome]`: Mostra somente detalhes sobre o usuário especificado.

---

## 2. last

- **Descrição**: Mostra uma listagem de entrada e saída de usuários no sistema.
- **Uso**:

```
bashCopiar código
last [opções]
```

- **Opções**:
  - `R`: Não mostra o campo HostName.
  - `a`: Mostra o hostname na última coluna. É útil se combinada com a opção `d`.
  - `d`: Usa o DNS para resolver o IP de sistemas remotos para nomes DNS.

---

## 3. logname

- **Descrição**: Mostra o login (username) do usuário atual.
- **Uso**:

```
logname
```

---

## 4. chfn

- **Descrição**: Muda os dados inseridos durante o cadastro do novo usuário.
- **Uso**:

```
chfn [opções] [usuário]
```

- **Onde**:
  - `usuário`: Nome do usuário para o qual os dados serão alterados.
- **Opções**:
  - `f [nome]`: Adiciona/altera o nome completo do usuário.
  - `r [nome]`: Adiciona/altera o número da sala do usuário.
  - `w [tel]`: Adiciona/altera o telefone de trabalho do usuário.
  - `h [tel]`: Adiciona/altera o telefone residencial do usuário.
  - `o [outros]`: Adiciona/altera outros dados do usuário.

## Adicionando Usuários

- `adduser [opções] [usuário/grupo]`

```
Adicionar usuário:

    adduser: Adiciona um novo usuário

Adicionar grupo:

    addgroup: Adiciona um novo grupo com as opções especif

Adicionar usuário a um grupo:

    adduser -a usuário grupo: Adiciona um usuário existent

Comandos para verificar usuários:

   Entra/muda de usuario: su nome_do_usuário: Entra no she
   Sair de usuario: logoutid : verifica o id do usuário
```

```
        id : verifica o id do usuário
        cat /etc/passwd | cut -d: -f1: ou cat /etc/group E
```

```
OPÇÕES COMUNS:

    -- hfn alfredo: para alterar as informações de contato
```

EXEMPLOS

- Para ver os usuários do sistema, digite `users`
- para logar com um usuário `su - alfredo`

```
#Adicionar um novo usuário:
adduser novo_usuario


#Adicionar um usuário existente a um grupo existente:

adduser usuario_existente grupo_existente


#Desativa verificações rigorosas de validade de nomes.
Exemplo:
adduser --allow-bad-names novo_usuario


#Define um comentário para a nova entrada gerada.
Exemplo:

adduser --comment "Novo usuário para departamento de venda


--gid:
```

```
Define o GID para o novo grupo ou grupo primário do novo u
Exemplo:



adduser --gid 1000 novo_usuario
```

**Removendo Usuários**

```
#Remover usuário do grupo


deluser ou userdel


Remover um usuário sem excluir o diretório home:

deluser --quiet joao


Remover um usuário e excluir o diretório home:

deluser --remove-home pedro



Remover um usuário e fazer backup dos arquivos antes

deluser --backup --backup-to /backup pedro

Remover um usuário do sistema:

deluser --system maria
```

```
Remover um grupo:

delgroup --verbose desenvolvedores


Remover um grupo apenas se estiver vazio:

delgroup --only-if-empty administradores


Remover um usuário de um grupo específico:

deluser --quiet ana administradores
```

**Logando em outro Grupos**

- para mudar o grupo principal da sessão atual para o grupo especificado

```
newgrp nome_do_grupo
```

**Adicionando Grupos**

- Esse diretório está em 'cat /etc/group' e tem as senhas no 'cat /etc/gshadow'
    - A divisão do diretório '/etc/group' se baseia em:
        - Ex: vboxusers:x:136:
        1. nome do grupo
        2. a senha está em outro arquivo
        3. identificação do grupo (gid)

```

#comando que faz você se pertencer a um grupo existente po
groupadd nome_do_grupo

#comando que faz você se pertencer a um grupo existente po
```

```
newgrp
exemplo: newgrp - strigus



-f (Indicador force):

#Este comando cria um novo grupo chamado "mygroup", mesmo
groupadd -f mygroup
```

**Removendo Grupos**

- você precisará remover os usuários do grupo ou transferi-los para outro grupo antes de tentar remover o grupo novamente.

```
delgroup ou grupdel



--only-if-empty: Remove um grupo apenas se estiver vazio.
Exemplo:

deluser --group --only-if-empty grupo

--remove-home: Remove o diretório home do usuário quando o
Exemplo:

deluser --remove-home usuário

--verbose: Fornece informações detalhadas durante a execuç
Exemplo:


deluser --verbose usuário
```

# Gerenciando senhas de Usuários

- No usuário normal, ele pede a senha atual pra modificar. No root não
- Quando é usuario, ele pede pra que a senha seja um pouco melhor.Já como root não precisa

```
#Para mudar a senha
passwd nome_usuario

#Dá uma senha para um grupo
gpasswd nome_do_grupo

#Caso o usuário esqueça a senha (-d de delete e -e de expi
passwd -de

#Deleta a senha para a conta.

passwd -d usuário

#Expira a senha para a conta.Agora vai pedir uma nova senh
passwd -e usuário




#muda a senha somente se estiver expirada.
passwd -k usuário


#tempo de conta inativa
passwd -i 30


# A senha fica inativa após a expiração
sudo chage -I 0 alfredo


#trava a conta
passwd -l usuário
```

```
# O usuário deve aguardar 3 dias para alterar a senha após
sudo chage -m 3 alfredo


# A senha deve ser alterada a cada 60 dias
sudo chage -M 60 alfredo


# Notificar sobre a expiração da senha com 5 dias de antec
sudo chage -W 5 alfredo


#Este comando destrava a conta indicada.
passwd -u usuário


#Após terminar o prazo, a senha deverá ser modificada

passwd -x MAX_DIAS

Exemplo:

passwd -x 90
```

## /etc/login.defs

- O arquivo `/etc/login.defs` é um arquivo de configuração em sistemas Linux que define as políticas e parâmetros para o comportamento de logins e gerenciamento de contas de usuários. Esse arquivo é utilizado por diversos comandos relacionados à gestão de usuários, como `useradd`, `userdel`, `usermod`, e outros.

### Principais Diretrizes e Configurações do `/etc/login.defs`

1. **Definição de UID e GID**:

- **UID_MIN** e **UID_MAX**: Definem o intervalo de IDs de usuários para contas de usuário normais (não-sistema).
- **GID_MIN** e **GID_MAX**: Definem o intervalo de IDs de grupo para grupos normais (não-sistema).

2. **Configurações de Expiração de Senha**:
   - **PASS_MAX_DAYS**: Número máximo de dias que uma senha pode ser usada antes de ser necessário alterá-la.
   - **PASS_MIN_DAYS**: Número mínimo de dias que uma senha deve ser mantida antes que possa ser alterada.
   - **PASS_WARN_AGE**: Número de dias para alertar o usuário antes da expiração da senha.
3. **Configurações de Diretório Home**:
   - **CREATE_HOME**: Define se um diretório home deve ser criado automaticamente ao adicionar um novo usuário.
   - **UMASK**: Define a máscara padrão para a criação de novos arquivos e diretórios.
4. **Segurança de Senhas**:
   - **ENCRYPT_METHOD**: Especifica o método de criptografia para senhas, como SHA512, MD5, etc.
   - **FAIL_DELAY**: Define o atraso em segundos após uma falha de login, como uma medida de segurança contra ataques de força bruta.
5. **Políticas de Shell e Login**:
   - **LOGIN_RETRIES**: Define o número de tentativas de login permitidas antes de uma ação ser tomada.
   - **LOGIN_TIMEOUT**: Tempo em segundos que o sistema espera para um login ser completado antes de ser cancelado.
   - **MAIL_DIR**: Diretório onde as caixas de correio dos usuários estão localizadas.
6. **Configurações Adicionais**:
   - **USERGROUPS_ENAB**: Define se cada usuário terá seu próprio grupo com o mesmo nome.

- **CHFN_RESTRICT**: Restringe quais campos de informações do usuário podem ser alterados pelo comando `chfn`.

```
Intervalo de UIDs para usuários normais
```

```
UID_MIN                  1000
UID_MAX                  60000
```

```
Intervalo de GIDs para grupos normais
```

```
GID_MIN                  1000
GID_MAX                  60000
```

```
Políticas de senha
```

```
PASS_MAX_DAYS            90
PASS_MIN_DAYS            0
PASS_WARN_AGE            7
```

```
Método de criptografia de senhas
```

```
ENCRYPT_METHOD           SHA512
```

```
Configurações de segurança
```

```
LOGIN_RETRIES            5
LOGIN_TIMEOUT            60
FAIL_DELAY               4
```

`chage` é utilizado em sistemas Linux para alterar e visualizar as informações de expiração de senha de um usuário. Ele permite definir políticas como a

validade da senha, o tempo de aviso antes da expiração e a data da última alteração. Isso é útil para forçar os usuários a alterarem suas senhas regularmente, contribuindo para a segurança do sistema.

## Comando `chage`

```
chage [opções] [usuário]
```

## Principais Opções do `chage`

- **l, --list**: Mostra as informações de expiração da senha do usuário.

  ```
  chage -l joao  # Exibe as configurações de expiração de senha do usuário 'joao'
  ```

- **m, --mindays [dias]**: Define o número mínimo de dias que a senha deve ser usada antes de ser alterada.

  ```
  chage -m 7 joao  # Define que a senha de 'joao' deve ser usada por pelo menos 7 dias antes de poder ser alterada.
  ```

- **M, --maxdays [dias]**: Define o número máximo de dias que a senha pode ser usada antes de expirar.

  ```
  bashCopiar código
  chage -M 90 joao  # Define que a senha de 'joao' expira após 90 dias.
  ```

- **W, --warndays [dias]**: Define o número de dias de aviso antes que a senha expire.

```
bashCopiar código
chage -W 7 joao  # O sistema avisará 'joao' 7 dias an
tes de a senha expirar.
```

- **I, --inactive [dias]**: Define o número de dias após a expiração da senha durante os quais a conta permanece ativa antes de ser desativada.

```
bashCopiar código
chage -I 30 joao  # A conta de 'joao' será desativada
30 dias após a expiração da senha, se não for alterad
a.
```

- **E, --expiredate [data]**: Define a data de expiração da conta do usuário.

```
bashCopiar código
chage -E 2024-12-31 joao  # Define que a conta de 'jo
ao' expira em 31 de dezembro de 2024.
```

- **d, --lastday [data]**: Define a data da última alteração da senha.

```
bashCopiar código
chage -d 2024-01-01 joao  # Define que a última alter
ação de senha de 'joao' foi em 1º de janeiro de 2024.
```

## Exemplo de Uso

Para listar as configurações de expiração de senha do usuário `joao`, você usaria:

```
bashCopiar código
chage -l joao
```

### Adicionando Usuários a determinados grupos

```
#Adiciona o usuario a um grupo
sudo usermod -G nerds alfredo

#Faz com que o usuário faça parte do grupo para um determi
su - nome_do_grupo     # para entrar no grupo (com base na
newgrp - nome_do_grupo # vai pedir a senha para entrar no

Faz com que o usuário faça parte do grupo
gpasswd -a nome_do_usuario nome_do_grupo #para entrar no g

Adiciona um 'lider' ao grupo
gpasswd -A nome_do_usuario_lider nome_do_grupo

Adiciona um 'lider' ao grupo e como 'membro' do grupo
gpasswd -M nome_do_usuario_lider nome_do_usuario_membro no
```

### Modificando os Usuários

- Para modificar, usa-se o `usermod`

```
  #adicionando o usuario a um grupo
sudo  usermod -a -G sudo joao


#Define o novo valor do ID do usuário.
  Exemplo: usermod -u 1001 username

  -g, --gid: Define o novo valor do ID do grupo

  Exemplo: usermod -u 5555 -g 0
```

```
  -c: Define um comentário para o grupo
    Exemplo: usermod -u 5555 -g 0 -c "giropops strigus"

    -s, --shell SHELL: Altera o shell de login do usuário.
   para alterar o shell, basta dar um (chsh lua)
    Exemplo: usermod -s /bin/bash girus

  -l: altera o nome de login de um usuário sem afetar outr
    Exemplo: usermod -l newname oldname


   -m, moveria o diretório home do usuário para /novo/dire
    usermod -m -d /novo/diretorio usuário
        Exemplo: usermod -m -d /home/strigus_novo strigus

 -b, --badnames: Permitir nomes não convencionais para as


        sudo usermod -b username


-e, --expiredate EXPIRE_DATE: Definir a data de expiração


        sudo usermod -e 2024-12-31 username

-f, --inactive INACTIVE: Definir o número de dias após a e



        sudo usermod -f 30 username

-g, --gid GROUP: Definir um novo grupo primário para a con

bash

        sudo usermod -g newgroup username
```

```
-G, --groups GROUPS: Definir uma nova lista de grupos supl

bash

        sudo usermod -G group1,group2 username

-a, --append: Adicionar o usuário a grupos suplementares s



        sudo usermod -a -G newgroup username

-l, --login NEW_LOGIN: Definir um novo nome de login para



        sudo usermod -l newlogin username

-L, --lock: Bloquear a conta de usuário.



        sudo usermod -L username

-m, --move-home: Mover o conteúdo do diretório home para u

        sudo usermod -m -d /new/home/dir username

-o, --non-unique: Permitir o uso de IDs de usuário duplica

        sudo usermod -o -u 1001 username

-p, --password PASSWORD: Definir uma nova senha criptograf

bash
```

```
        sudo usermod -p '$6$random_salt$encrypted_password

-s, --shell SHELL: Definir um novo shell de login para a c

bash

        sudo usermod -s /bin/bash username

-u, --uid UID: Definir um novo UID (identificador de usuár


        sudo usermod -u 1002 username

-U, --unlock: Destravar a conta de usuário.

        sudo usermod -U username

-v, --add-subuids FIRST-LAST: Adicionar um intervalo de UI


        sudo usermod -v 10000-20000 username

-V, --del-subuids FIRST-LAST: Remover um intervalo de UID


        sudo usermod -V 10000-20000 username

-w, --add-subgids FIRST-LAST: Adicionar um intervalo de GI


        sudo usermod -w 10000-20000 username

-W, --del-subgids FIRST-LAST: Remover um intervalo de GID


        sudo usermod -W 10000-20000 username

-Z, --selinux-user SEUSER: Definir um novo mapeamento de u
```

```
        sudo usermod -Z system_u username


- `chfn` é usado para alterar as informações de comentário

`chfn [opções] [nome_do_usuário]`

Por exemplo, para alterar o nome completo de um usuário ch
chfn -f "John Doe" johndoe
```

### Modificando os Grupos

- Para modificar, usa-se o `groupmod`

```
gpasswd nome_do_grupo
Dá uma senha para um grupo


-g, --gid GID: Alterar o ID do grupo para um novo valor.


sudo groupmod -g 1001 groupname

-h, --help: Exibir mensagem de ajuda e sair.


groupmod -h

-n, --new-name NEW_GROUP: Alterar o nome do grupo para um


sudo groupmod -n newgroupname groupname

-o, --non-unique: Permitir o uso de IDs de grupo duplicado
```

```
sudo groupmod -o -g 1001 groupname

-p, --password PASSWORD: Alterar a senha do grupo para uma

sudo groupmod -p '$6$random_salt$encrypted_password' group

-R, --root CHROOT_DIR: Especificar um diretório raiz para a

sudo groupmod -R /newroot groupname

-P, --prefix PREFIX_DIR: Especificar um diretório prefixo

sudo groupmod -P /newprefix groupname
```

# gpasswd

- Utilizado pra administrar grupos em sistemas Linux. Ele permite adicionar ou remover usuários de grupos, definir senhas para grupos e configurar o administrador de um grupo. Isso é especialmente útil para gerenciar permissões de acesso em um ambiente multiusuário.

## Sintaxe do Comando `gpasswd`

```
gpasswd [opções] [grupo]
```

```
# Adiciona o usuário 'joao' ao grupo 'devs'.
```

```
gpasswd -a joao devs

#Definir Senha para um Grupo
sudo gpasswd nerds

#Adicionar um Usuário como Administrador do Grupo
sudo gpasswd -A alfredo nerds

#Comando para Remover um Grupo
delgroup nome_do_grupo

#Remove 'ana' do grupo 'devs'.
gpasswd -d ana devs
```

## Principais Opções do `gpasswd`

- **a, --add [usuário]**: Adiciona um usuário a um grupo.

```
gpasswd -a joao devs  # Adiciona o usuário 'joao' ao grupo 'devs'.
```

- **d, --delete [usuário]**: Remove um usuário de um grupo.

```
gpasswd -d joao devs  # Remove o usuário 'joao' do grupo 'devs'.
```

- **A, --administrators [usuários]**: Define administradores do grupo.

```
gpasswd -A maria,paulo devs  # Define 'maria' e 'paulo' como administradores do grupo 'devs'.
```

- **M, --members [usuários]**: Define os membros do grupo, substituindo a lista atual.

```
gpasswd -M joao,ana,paulo devs  # Define 'joao', 'an
a' e 'paulo' como os membros do grupo 'devs'.
```

- **[grupo]** (sem opções): Solicita a definição de uma senha para o grupo. Essa senha pode ser usada para acesso temporário ao grupo através do comando `newgrp`.

```
gpasswd devs  # Define uma senha para o grupo 'devs'.
```

## Exemplos de Uso

1. **Adicionar um usuário a um grupo:**

```
gpasswd -a joao sudo  # Adiciona 'joao' ao grupo 'sud
o'.
```

2. **Remover um usuário de um grupo:**

```
gpasswd -d ana devs  # Remove 'ana' do grupo 'devs'.
```

3. **Definir administradores de um grupo:**

```
gpasswd -A maria devs  # Define 'maria' como administ
radora do grupo 'devs'.
```

4. **Definir membros de um grupo (substitui membros atuais):**

```
gpasswd -M joao,ana,paulo devs  # Define 'joao', 'an
a', e 'paulo' como os únicos membros do grupo 'devs'.
```

5. **Definir uma senha para o grupo:**

```
gpasswd devs  # Define uma senha para o grupo 'devs'.
```

▼ ***Find e Grep***

# Comandos `find`

- **Caminho**:
  Especifica o diretório inicial onde o
  `find` começa a busca. Pode ser um único diretório, ou vários, separados por espaço. Exemplo: `find /home/user`.
- **Opções**:
  Ajusta como o
  `find` realiza a busca. Algumas das mais usadas são:
  - `name`: Busca por nome (aceita curingas `*`, como em `name "*.txt"`).
  - `iname`: Busca por nome, ignorando maiúsculas e minúsculas.
  - `type`: Filtra o tipo de arquivo, como:
    - `f` para arquivos,
    - `d` para diretórios,
    - `l` para links simbólicos.
  - `size`: Busca por tamanho de arquivo (ex: `size +100k` para arquivos maiores que 100 KB).
  - `mtime`: Encontra arquivos modificados nos últimos N dias (ex: `mtime -7` para modificações nos últimos 7 dias).

- `user`: Busca arquivos pertencentes a um usuário específico (ex: `user root`).
- `perm`: Filtra por permissão de arquivo (ex: `perm 755`).

- **Condições**:
  Define critérios adicionais para filtrar arquivos, podendo usar expressões lógicas:
  - `and`: Condição lógica "E".
  - `or`: Condição lógica "OU".
  - `not`: Inverte a condição (ex: `not -name "*.log"` encontra tudo exceto arquivos `.log`).
- **Ações**:
  Define o que o `find` faz com cada arquivo encontrado.
  - `print`: Exibe o caminho do arquivo (ação padrão).
  - `exec [comando] {}`: Executa um comando em cada arquivo encontrado (ex: `exec rm {} \;` para remover os arquivos encontrados).
  - `delete`: Remove os arquivos encontrados (cuidado ao usar).
- **Exemplo de Pesquisa de Diretórios e Arquivos**:

```
find /home/samuel -name "a*" -type d           # Busc
a diretórios que começam com "a"
find /home/samuel -ctime +1                    # Arqu
ivos criados há mais de um dia
find /home/samuel -name "a*" -exec cp {} dir/ \; # bu
sca todos os arquivos e diretórios no diretório /hom
e/samuel (e subdiretórios) cujos nomes começam com
"a" e copia-os para o diretório dir/.O exec permite e
xecutar um comando específico em cada arquivo ou dire
tório que o find encontra
```

- `du -hs` mostrará o tamanho total
- `find / -name "arquivo.txt"`
  - Procura por arquivos e diretórios chamados "arquivo.txt" a partir do diretório raiz `/`.
- `find . -name nome_do_arquivo`
  - Procura por arquivos e diretórios com o nome especificado a partir do diretório atual (`.`).
- `find . -type d -name mc`
  - Encontra apenas diretórios chamados "mc" a partir do diretório atual.
- `find . -type f -name mc`
  - Encontra apenas arquivos chamados "mc" a partir do diretório atual.
- `find .local -type f -name *gname*`
  - Encontra dentro de .local todos os arquivos que tem gname no meio
- `find /usr -maxdepth 2 -type f -name mc`
  - Procura por arquivos chamados "mc" dentro do diretório `/usr`, mas apenas até dois níveis de subdiretórios.
- `find . -mtime -1`
  - Lista arquivos que foram modificados no último dia.
- `find . -ctime -1`
  - Lista arquivos que foram criados no último dia.
- `find . -atime -1`
  - Lista arquivos que foram acessados no último dia.
- `find /usr -size +1000`
  - Lista arquivos dentro do diretório `/usr` que têm mais de 1000 bytes.

▼ *Comandos Avançados*

# Editor de Texto `ed`

O `ed` é um editor de texto orientado a linhas, ideal para edições rápidas ou automáticas em arquivos.

## Abrir um arquivo no `ed`

```
bash
Copiar código
ed <nome_do_arquivo>
```

---

## Comandos básicos dentro do `ed`

- `a`: Insere texto **após** a linha atual.
- `i`: Insere texto **antes** da linha atual.
- `d`: Deleta a(s) linha(s) especificada(s).
- `,p`: Imprime **todas as linhas**.
- `q`: Sai do editor.
  - `q!`: Sai **sem salvar** alterações.
- `w`: Salva o arquivo.
  - `:w`: Mostra a quantidade de bytes salvos.
- `g`
  - Para substituir todas as ocorrências de "velho" por "novo" na linha atual:

```
s/velho/novo/g
```

- Para substituir todas as ocorrências no arquivo inteiro (comando global no `vim`, por exemplo):

```
:%s/velho/novo/g
```

---

## Imprimir linhas específicas

- `1,$ p`: Imprime o texto inteiro.Substitua `1,$` por:
  - `2,&`: Para imprimir um intervalo ou linha específica.

---

## Exibir números de linha

```
:1,$ n
```

---

## Localizar uma palavra

- `/palavra`: Localiza a **próxima ocorrência** da palavra especificada.
  - Após encontrar, você pode usar:
    - `n`: Para ir para a **próxima ocorrência**.
    - `?`: Para buscar **ocorrências anteriores**.

---

## Modificar uma linha específica

1. Vá para a linha com:

```
:3
```

2. Substitua o conteúdo com:

```
c
Tecnologia de Redes de Computadores
```

```
.
```

---

### Substituir texto ( `s` )

O comando `s` substitui texto em linhas específicas ou no arquivo todo.

### Sintaxe

```
s/<padrão>/<novo_texto>/
```

### Exemplos

- **Substituir a primeira ocorrência** de "velho" por "novo" na linha atual:

```
s/velho/novo/
```

- **Substituir todas as ocorrências** na linha atual:

```
s/velho/novo/g
```

---

### Dicas Gerais

- Para operações em todas as linhas do arquivo, use `1,$` no lugar do número da linha.
- Salve alterações frequentemente com `w`.
- Saia sem medo de perder o conteúdo com `q!`.

## Comando: `EOF`

**Descrição**: EOF significa *End of File* (fim do arquivo). É um indicador usado em vários contextos, principalmente para sinalizar o término de entrada de dados

### Criar um arquivo com conteúdo usando EOF

```
cat <<EOF > arquivo.txt
Este é um exemplo de texto.
Tudo o que estiver aqui será salvo no arquivo.
EOF
```

- `<<EOF`: Inicia o bloco de texto.
- `EOF`: Indica o fim do bloco.

# COMANDO SED

### Sintaxe básica

```
sed [opções] 'comando' arquivo



#IMPRIMINDO:
#Imprimir as linhas 2 a 4:
#
#sed -n '2,4p' arquivo.txt


#ADICIONAR UM TEXTO EM X LINHA
#
#sed '3a \Teste de Inserção' arquivo01.txt
```

### Comandos comuns no `sed`

- `s/padrão/substituição/`: Substitui a primeira ocorrência do padrão por um texto de substituição na linha atual.
  - Exemplo: Substituir "velho" por "novo" na linha:

```
sed 's/velho/novo/' arquivo.txt
```

- `s/padrão/substituição/g`: Substitui **todas** as ocorrências do padrão na linha.
  - Exemplo: Substituir todas as ocorrências de "velho" por "novo" em cada linha:

```
sed 's/velho/novo/g' arquivo.txt
```

- `s/padrão/substituição/numero`: Substitui a N-ésima ocorrência de um padrão em uma linha.
  - Exemplo: Substituir a segunda ocorrência de "velho" por "novo":

```
sed 's/velho/novo/2' arquivo.txt
```

- `d`: Deleta a(s) linha(s) que correspondem ao padrão.
  - Exemplo: Deletar linhas que contenham "erro":

```
sed '/erro/d' arquivo.txt
```

- `p`: Imprime as linhas que correspondem ao padrão. Usado em conjunto com a opção `n` para evitar a impressão de todas as linhas.
  - Exemplo: Imprimir apenas as linhas que contêm "sucesso":

```
sed -n '/sucesso/p' arquivo.txt
```

- `i`: Insere uma linha antes da linha que corresponde ao padrão.
  - Exemplo: Inserir uma linha antes da linha que contém "erro":

```
sed '/erro/i Nova linha antes do erro' arquivo.txt
```

- `a`: Adiciona uma linha após a linha que corresponde ao padrão.
  - Exemplo: Adicionar uma linha após a linha que contém "sucesso":

```
sed '/sucesso/a Nova linha após o sucesso' arquivo.txt
```

---

## Substituições avançadas no `sed`

- **Substituir no arquivo**: Para salvar as alterações diretamente no arquivo (sem exibir na tela), use a opção `i` (in-place):

```
sed -i 's/velho/novo/g' arquivo.txt
```

- **Substituir no arquivo, mas com backup**: Para fazer a substituição e ainda salvar um backup do arquivo original, adicione uma extensão para o backup:

```
sed -i.bak 's/velho/novo/g' arquivo.txt
```

- **Substituir com uma expressão regular**: O `sed` suporta expressões regulares para padrões mais complexos.
    - Exemplo: Substituir números de telefone no formato "(xx) xxxx-xxxx" para "xxxx-xxxx":

```
sed 's/(\([0-9]*\)) \([0-9]*\)-\([0-9]*\)/\2-\3/' arquivo.txt
```

- **modificando-o** sem precisar redirecionar a saída para um novo arquivo. Quando você usa `-i`, o `sed` aplica as alterações diretamente no arquivo original.

### Exemplo:

Para substituir todas as ocorrências de "erro" por "sucesso" no arquivo `arquivo.txt`, você pode usar o seguinte comando com a opção `-i`:

```
sed -i 's/erro/sucesso/' arquivo.txt
```

- `r` **(Extended Regular Expressions)**

A opção `-r` permite usar **expressões regulares estendidas**

### Exemplo:

Se você quiser substituir a primeira ocorrência de "erro" ou "falha" por "sucesso", pode usar o seguinte comando com `-r`:

```
sed -r 's/(erro|falha)/sucesso/' arquivo.txt
```

# CUT

*`cut [opções] [arquivo]`*

- `f`: Especifica os campos a serem extraídos. Você deve usar em conjunto com a opção `d` para definir o delimitador.
  - Exemplo: `cut -f1,3 arquivo.txt` extrai o 1º e 3º campos do arquivo.
- `d`: Define o delimitador de campo (o caractere que separa os campos).
  - Exemplo: `cut -d',' -f2 arquivo.csv` usa a vírgula como delimitador e extrai o 2º campo.
- `c`: Especifica as posições de caracteres a serem extraídos.
  - Exemplo: `cut -c1-5 arquivo.txt` extrai os caracteres da posição 1 a 5 de cada linha.
- `s`: Supressão de linhas que não contêm o delimitador.
  - Exemplo: `cut -d',' -f1 -s arquivo.txt` só mostra linhas que contêm uma vírgula.

## *CORTANDO CAMPOS*

`cut -d ":" -f 1 /etc/passwd` - pega o primeiro campo

`cut -d ":" -f 1, 2 /etc/passwd` - pega o primeiro e segundo campo

`cut -b 1-4 /etc/passwd` - pega os bytes de 1 ao 4, pega todos os espaços

`cut -c 1-4 /etc/passwd` - pega os bytes de 1 ao 4, porém sem contar os espaços

## `tr` para substituições simples de caracteres

Se você está tentando substituir caracteres individuais de "abcde" para "ABCDE" (e não uma string específica como "abcde"), você pode usar o comando `tr`:

```
bash
Copiar código
```

```
cat arquivo01.txt | tr 'abcde' 'ABCDE'
```

### Remover caracteres específicos

Para remover caracteres de uma cadeia de texto, use a opção `-d`. Por exemplo, para remover todas as letras "a" de uma string:

```
echo "banana" | tr -d 'a'
```

# UNIQUE

- `c`: Conta e exibe o número de ocorrências de cada linha.
- Se você tem um arquivo ou uma lista de texto com linhas duplicadas consecutivas e deseja eliminar as repetições, basta usar:

```
uniq arquivo.txt
```

### Ignorar diferenças de maiúsculas e minúsculas

Para tratar "apple" e "Apple" como a mesma linha (ignorando diferenças de maiúsculas e minúsculas), use a opção `-i`:

```
uniq -i arquivo.txt
```

### Exibir apenas as linhas duplicadas

Se você deseja ver apenas as linhas que são duplicadas, use a opção `-d`:

```
uniq -d arquivo.txt
```

# EXPR ou BC ou $(( ))

- Formas para calcular

```
#EXPR
expr 5 + 3


#BC
echo "1+2+3+5" | bc

#Usando $(()) para fazer calculo
echo $((1+2))

#Escala para especificar a quantidade de casa
echo "scale=2;5.0/2" | bc
```

▼ **Variáveis e Parâmetros**

## 1. Variáveis

## Definindo e usando variáveis

- Atribuição: `nome_variavel=valor`
  - Sem espaços ao redor do `=`.
- Para acessar o valor de uma variável: prefixe o nome com `$`.

**Exemplo:**

```
#!/bin/bash
# Definindo variáveis
nome="Allyson"
idade=25

# Usando as variáveis
echo "Nome: $nome"
```

```
echo "Idade: $idade"

#nao se pode fazer:
variavel = "Casa amarela"
variavel ="Casa amarela"
```

## Parâmetros Posicionais

- `$0`: Nome do script.
- `$1`, `$2`, ..., `$n`: Parâmetros passados na chamada.
- `$#`: Número total de parâmetros passados.
- `$*`: Todos os parâmetros como uma string única.
- `$@`: Todos os parâmetros como uma lista.

**Exemplo:**

```
#!/bin/bash
echo "Nome do script: $0"
echo "Primeiro parâmetro: $1"
echo "Segundo parâmetro: $2"
echo "Número de parâmetros: $#"
echo "Todos os parâmetros: $*"
```

**Execução:**

```
./meu_script.sh "Allyson" "Freire"
```

**Saída:**

```
Nome do script: ./meu_script.sh
Primeiro parâmetro: Allyson
```

```
Segundo parâmetro: Freire
Número de parâmetros: 2
Todos os parâmetros: Allyson Freire
```

## `xargs`

- O `xargs` pega a entrada (separada por **espaços** ou **linhas**) e agrupa os itens como argumentos para o comando.
  - **Sem `xargs`:** Você precisa copiar a saída manualmente e passá-la para o próximo comando.
  - **Com `xargs`:** Todo o processo é automatizado; o `xargs` faz a ponte entre os comandos.

    Com a opção `-I`, você pode dizer ao `xargs` onde colocar os argumentos.

    **Exemplo: Mover arquivos**

    ```
    echo "file1 file2 file3" | xargs -I {} mv {} /destino
    ```

    ### 1. Sem usar `xargs`:

    1. Liste os arquivos `.log` usando o comando `find`:

       ```
       find . -name "*.log"
       ```

       **Saída:**

       ```
       ./arquivo1.log
       ./subdir/arquivo2.log
       ```

```
./subdir2/arquivo3.log
```

1. Agora, copie e cole manualmente a lista no comando `rm`:

   ```
   rm ./arquivo1.log ./subdir/arquivo2.log ./su
   bdir2/arquivo3.log
   ```

   Isso funciona, mas é **manual** e fica impraticável se houver muitos arquivos.

### 2. Usando `xargs`:

1. Combinamos o comando `find` com `xargs`:

   ```
   find . -name "*.log" | xargs rm
   ```

▼ **Agrupando condições**

- **Combinando Números e Strings**

  Se quiser misturar comparações de números e strings

```
[ \( 10 -eq 10 \) -o \( 20 -eq 15 \) ] && echo "Uma condiç
```