

Curso de LPIC-1

Tópico 103.2 – Aplicando Filtro a Textos e Arquivos

**cat, tac, head, tail, less, wc, nl, sort, uniq, od, join, paste, split, tr,
cut, sed, xzcat, bzcat, zcat, md5sum, sha256sum, sha512sum**

Douglas Luna - @dougluna 

O que estudamos nesse tópico:

cat.....	3
tac.....	6
head.....	7
tail.....	9
less	12
wc	14
nl.....	16
sort.....	17
uniq.....	19
od.....	22
join	24
paste.....	25
split	26
tr	28
cut.....	33
sed.....	37
xzcat	41
bzcat.....	42
zcat.....	43
Comandos de checksum.....	44
md5sum	44
sha256sum.....	45
sha512sum.....	46

cat

O comando **cat** lê e concatena um arquivo. Ao executar o comando ele lê o conteúdo desse arquivo e printa na saída padrão, que é a sua tela.

Por exemplo temos um arquivo chamado alunos.txt:

Sintaxe:

```
$cat <ARQUIVO>
```

```
$cat alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ cat alunos.txt
Andre Gonçalves
Paulo Freitas
Maria Antonieto Sousa

Carlos Augusto
Ana Claudia
Ana Claudia Vasconcelos Ana

Rafael dos Santos
Silvia Oliveira
Antonio      Silva
Eliseu       Padilha
Ricardo

dougluna@dip:~/lpi/Exercicios$
```

Opções interessantes do cat

```
$cat -n <ARQUIVO>
```

-n → Que mostra todas as linhas numeradas, cada linha e seu número.

```
dougluna@dip:~/lpi/Exercicios$ cat -n alunos.txt
1 Andre Gonçalves
2 Paulo Freitas
3 Maria Antonieto Sousa
4
5 Carlos Augusto
6 Ana Claudia
7 Ana Claudia Vasconcelos Ana
8
9
10 Rafael dos Santos
11 Silvia Oliveira
12 Antonio      Silva
13 Eliseu       Padilha
14 Ricardo

dougluna@dip:~/lpi/Exercicios$ |
```

```
$cat -b <ARQUIVO>
```

-b → Que mostra apenas as linhas não brancas numeradas, aquelas que tem conteúdo.

```
dougluna@dip:~/lpi/Exercicios$ cat -b alunos.txt
 1 Andre Gonçalves
 2 Paulo Freitas
 3 Maria Antonieto Sousa

 4 Carlos Augusto
 5 Ana Claudia
 6 Ana Claudia Vasconcelos Ana

 7 Rafael dos Santos
 8 Silvia Oliveira
 9 Antonio           Silva
10 Eliseu            Padilha
11 Ricardo

dougluna@dip:~/lpi/Exercicios$
```

```
$cat -s <ARQUIVO>
```

-s → Quando ele tem mais de uma linha em branco de forma sequencial, ele corta essa sequência e mantém no máximo 1 linha em branco.

```
dougluna@dip:~/lpi/Exercicios$ cat -s alunos.txt
Andre Gonçalves
Paulo Freitas
Maria Antonieto Sousa

Carlos Augusto
Ana Claudia
Ana Claudia Vasconcelos Ana

Rafael dos Santos
Silvia Oliveira
Antonio           Silva
Eliseu            Padilha
Ricardo

dougluna@dip:~/lpi/Exercicios$
```

```
$cat -A <ARQUIVO>
```

-A → Quando ele usado, podemos ver todos os caracteres especiais no conteúdo do arquivo, por exemplo todo fim de linha ele tem um cifrão, toda vez que tem um Tab ele mostra o “^I” que significa um Tab, quando tem um caractere especial ele mostra o código desse caractere.

```
dougluna@dip:~/lpi/Exercicios$ cat -A alunos.txt
Andre GonM-CM-'alves$
Paulo Freitas$
Maria Antonieto Sousa$
$
Carlos Augusto$
Ana Claudia$
Ana Claudia Vasconcelos Ana$
$
$
Rafael dos Santos$
Silvia Oliveira$
Antonio^I^ISilva$
Eliseu      Padilha$
Ricardo$
dougluna@dip:~/lpi/Exercicios$
```

tac

O comando **tac** faz o reverso do cat, ele lê arquivos e concatena ao contrário. **O tac cria uma cópia de cada arquivo e manda para saída padrão, revertendo as gravações do arquivo (linha por linha) e printa da última linha para a primeira.**

Sintaxe:

```
$tac <ARQUIVO>
```

```
$tac alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ tac alunos.txt
Ricardo
Eliseu      Padilha
Antonio     Silva
Silvia Oliveira
Rafael dos Santos

Ana Claudia Vasconcelos Ana
Ana Claudia
Carlos Augusto

Maria Antonieto Sousa
Paulo Freitas
Andre Gonçalves
dougluna@dip:~/lpi/Exercicios$
```

Para ver as opções do tac, use o:

```
$tac --help
```

```
dougluna@dip:~/lpi/Exercicios$ tac --help
Usage: tac [OPTION]... [FILE]...
Write each FILE to standard output, last line first.

With no FILE, or when FILE is -, read standard input.

Mandatory arguments to long options are mandatory for short options too.
  -b, --before           attach the separator before instead of after
  -r, --regex            interpret the separator as a regular expression
  -s, --separator=STRING use STRING as the separator instead of newline
  --help                display this help and exit
  --version             output version information and exit

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Report tac translation bugs to <https://translationproject.org/team/>
Full documentation at: <https://www.gnu.org/software/coreutils/tac>
or available locally via: info '(coreutils) tac invocation'
dougluna@dip:~/lpi/Exercicios$
```

head

O comando **head**, como o nome diz, mostra o cabeçalho do arquivo, ou seja, as primeiras linhas dele. Então se digitarmos o comando:

Sintaxe:

```
$ head <ARQUIVO>
```

```
$head alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ head alunos.txt
Andre Gonçalves
Paulo Freitas
Maria Antonieto Sousa

Carlos Augusto
Ana Claudia
Ana Claudia Vasconcelos Ana

Rafael dos Santos
dougluna@dip:~/lpi/Exercicios$
```

Como padrão, vemos o conteúdo das 10 primeiras linhas do arquivo.

Então como visto na imagem, o padrão é que ele mostra as primeiras 10 linhas e tem alguns Linux que o padrão pode ser 5 linhas.

Opção para definir o número de linhas:

```
$ head -n<NUMERO> <ARQUIVO>
```

-n<NUMERO> → Se quiser definir quantas linhas será exibida pode utilizar a opção **-n<NUMERO_DESEJADO>**.

```
dougluna@dip:~/lpi/Exercicios$ head -n2 arquivolongo.txt
exemplo do ZZ

dougluna@dip:~/lpi/Exercicios$
```

As 2 primeiras linhas foram exibidas com sucesso.

Opção para definir o número de linhas:

```
$ head -<NUMERO> <ARQUIVO>
```

-<NUMERO> → Também podemos fazer dessa forma se quisermos definir quantas linhas será exibida pode utilizar a opção -<NUMERO_DESEJADO>.

```
dougluna@dip:~/lpi/Exercicios$ head -2 arquivo longo.txt  
exemplo do ZZ
```

```
dougluna@dip:~/lpi/Exercicios$
```

As 2 primeiras linhas foram exibidas com sucesso.

Opção para definir o número de bytes que será exibido:

```
$ head -c<NUMERO> <ARQUIVO>
```

-c<NUMERO> → Para definir quantas bytes será exibido utilize a opção -c

```
dougluna@dip:~/lpi/Exercicios$ head -c55 arquivo longo.txt  
exemplo do ZZ
```

```
LPI Linux Essentials  
ooooasfafja  
aoaoaoao dougluna@dip:~/lpi/Exercicios$
```

Os 55 primeiros bytes foram exibidos com sucesso.

tail

O comando tail, é bem semelhante ao head, mas ele mostra as linhas finais do arquivo e tem os mesmos parâmetros. Vamos ver seu funcionamento:

Sintaxe:

```
$ tail <ARQUIVO>
```

```
$ tail arquivolongo.txt
```

```
dougluna@dip:~/lpi/Exercicios$ tail arquivolongo.txt
Languages: English, German, Portuguese (Brazilian), Japanese

To become LPIC-2 certified the candidate must be able to:

    perform advanced system administration, including common tasks regarding the Linux kernel, system
startup and maintenance;
    perform advanced Management of block storage and file systems as well as advanced networking and a
uthentication and system security, including firewall and VPN;
    install and configure fundamental network services, including DHCP, DNS, SSH, Web servers, file s
ervers using FTP, NFS and Samba, email delivery; and
    supervise assistants and advise management on automation and purchases.
```

```
dougluna@dip:~/lpi/Exercicios$
```

Como padrão, vemos o conteúdo das 10 últimas linhas do arquivo.

Então como visto na imagem, o padrão é que ele mostra as últimas 10 linhas e tem alguns Linux que o padrão pode ser 5 linhas.

Opção para definir o número de linhas que serão exibidas:

```
$ tail -n<NUMERO> <ARQUIVO>
```

Ou

```
$ tail -<NUMERO> <ARQUIVO>
```

```
dougluna@dip:~/lpi/Exercicios$ tail -n3 arquivolongo.txt
supervise assistants and advise management on automation and purchases.
```

```
dougluna@dip:~/lpi/Exercicios$ tail -3 arquivolongo.txt
supervise assistants and advise management on automation and purchases.
```

```
dougluna@dip:~/lpi/Exercicios$
```

Nesse caso, escolhemos as últimas 3 linhas do arquivo.

```
$ tail -f <ARQUIVO>
```

-f → O tail possui a opção **-f**, ao utilizarmos essa opção, ele vai ler o arquivo e ficará pendente, ou seja, esperando a entrada de dados. Se algo for escrito nesse arquivo enquanto ele estiver pendente poderemos ver ao vivo essa nova entrada.

Abrimos uma o arquivo alunos.txt com a opção **-f**:

```
$ tail -f alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ tail -f alunos.txt
Carlos Augusto
Ana Claudia
Ana Claudia Vasconcelos Ana

Rafael dos Santos
Silvia Oliveira
Antonio Silva
Eliseu Padilha
Ricardo
```

O arquivo está aberto e pendente.

Abrimos outro shell para inserir um novo valor no arquivo alunos.txt usando o redirecionamento, pegando a saída do comando echo e jogando para o final do arquivo alunos.txt:

```
$ echo "Douglas Luna" >> alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ echo "Douglas Luna" >> alunos.txt
dougluna@dip:~/lpi/Exercicios$ |
```

Redirecionamento executado com sucesso.

Ao retornamos para o shell onde o arquivo foi aberto com **tail -f** veremos a nova entrada:

```
dougluna@dip:~/lpi/Exercicios$ tail -f alunos.txt
Carlos Augusto
Ana Claudia
Ana Claudia Vasconcelos Ana

Rafael dos Santos
Silvia Oliveira
Antonio      Silva
Eliseu       Padilha
Ricardo
Douglas Luna
|
```

Podemos ver o valor Douglas Luna inserido.

Por fim, concluímos que **essa opção -f do tail é muito útil para verificar arquivos de logs, que possuem alta entrada de valores e facilitando a visualização ao vivo delas.**

Podemos combinar a opção **-n** com a **-f**. Exemplo vamos verificar as últimas 5 linhas ao mesmo tempo que mantemos o acompanhamento no arquivo alunos.txt:

```
$ tail -n5 -f alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ tail -n5 -f alunos.txt
Silvia Oliveira
Antonio      Silva
Eliseu       Padilha
Ricardo
Douglas Luna
|
```

less

O comando **less**, nos ajuda a fazer a paginação de um arquivo muito longo, facilitando a leitura e navegação no mesmo. O **less** é uma evolução do comando **more**, o **less** possui mais opções que o seu antecessor **more** e a LPÍ exige apenas o comando **less**.

Sintaxe:

```
$ less <ARQUIVO>
```

```
$ less arquivolongo.txt
```

```
exemplo do ZZ

LPI Linux Essentials
ooooasfafja
aoaoaoa
aaaa
Show employers that you have the foundational skills required for your next job or promotion.

Linux adoption continues to rise world-wide as individual users, government entities and industries ranging from automotive to space exploration embrace open source technologies. This expansion of open source in enterprise is redefining traditional Information and Communication Technology (ICT) job roles to require more Linux skills. Whether you're starting your career in open source, or looking for advancement, independently verifying your skill set can help you stand out to hiring managers or your management team.

The Linux Essentials Professional Development Certificate (PDC) also serves as an ideal stepping-stone to the more advanced LPIC Professional Certification track for Linux Systems Administrators.
aisafisa
kaklsaflkafjasjfafdf
Ricardo
Current Version: 1.5 (Exam code 010-150)

Prerequisites: There are no prerequisites for this certification
:|
[0] 0:less* 1:bash-                                     "dip" 20:29 12-Jun-20
```

Vemos a primeira página, e assim conseguimos navegador pelo arquivo com as setas e fazer buscas por palavras chaves.

Teclas das **setas para baixo ou para cima** e assim navegamos linha a linha.

Tecla Enter para ir navegamos linha a linha.

Barra de Espaço pula para próxima página.

/**PALAVRA** para buscar um valor no arquivo.

Exemplo:

```
/101
```

```
The world's largest and most recognized Linux Certification

/101
[0] 0:less* 1:bash-
```

```
Current Version: 4.0 (Exam codes 101-400 and 102-400)
Current Version: 4.0 (Exam codes 101-400 and 102-400)
Current Version: 4.0 (Exam codes 101-400 and 102-400)
Aula DE VI
Aula DE VI
Aula DE VI
```

O valor buscado foi encontrado.

Se apertamos **N** ele vai para o próximo valor da palavra que procuramos.

Shift+N volta para a palavra anterior daquilo que buscamos.

Ctrl+G ou **Ctrl+g** mostra o status do arquivo e onde estou nele:

```
have a basic understanding of security and administration relate
ment, working on the command line, and permissions.

Aula DE VI
Aula DE VI
Aula DE VI
arquivolongo.txt lines 23-40/93 byte 1963/4965 40% (press RETURN)
[0] 0:less* 1:bash-
```

Status do arquivo, estou em 40% dele.

O less também é muito usado com o Pipe “ | ” , onde pegamos a saída do primeiro comando e uso de entrada para o segundo comando. Então ao digitar **cat arquivolongo.txt | less** o less vai controlar a saída desse cat. Resumindo, vai funcionar da mesma maneira que se estiver digitando **less**.

```
$ cat arquivolongo.txt | less
```

```
dougluna@dip:~/lpi/Exercicios$ cat arquivolongo.txt | less
dougluna@dip:~/lpi/Exercicios$ |
```

```
Current Version: 1.5 (Exam code 010-150)
```

```
Prerequisites: There are no prerequisites for this certification
```

```
:|
```

```
[0] 0:bash* 1:bash-
```

WC

O comando **wc**, mostra a quantidade de linha, palavra e byte que possuí em um arquivo.

Sintaxe:

```
$ wc <ARQUIVO>
```

```
$ wc alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ wc alunos.txt  
15 27 204 alunos.txt  
dougluna@dip:~/lpi/Exercicios$
```

Linhas 15, palavras 27, bytes 204.

```
$ wc -l <ARQUIVO>
```

-l → A opção **-l (lines)** nós retorna apenas a quantidade de linhas do arquivo:

```
dougluna@dip:~/lpi/Exercicios$ wc -l alunos.txt  
15 alunos.txt  
dougluna@dip:~/lpi/Exercicios$ |
```

```
$ wc -w <ARQUIVO>
```

-W → A opção **-w (words)** nós retorna apenas a quantidade de palavras do arquivo:

```
dougluna@dip:~/lpi/Exercicios$ wc -w alunos.txt  
27 alunos.txt  
dougluna@dip:~/lpi/Exercicios$
```

```
$ wc -c <ARQUIVO>
```

-C → A opção **-c (bytes)** nós retorna apenas a quantidade de bytes do arquivo:

```
dougluna@dip:~/lpi/Exercicios$ wc -c alunos.txt  
204 alunos.txt  
dougluna@dip:~/lpi/Exercicios$
```

```
$ wc -m <ARQUIVO>
```

-M → A opção **-m (characters)** nós retorna apenas a quantidade de caracteres do arquivo

Podemos usar o wc para vários arquivos ao mesmo tempo. Estamos num diretório que possui vários arquivos então executaremos o comando:

```
$ wc *
```

```
dougluna@dip:~/lpi/Exercicios$ ls
Script_Exemplo.sh    alunos.txt      codigo-aluno.txt  texto.txt
Script_Variavel.sh   arquivolongo.txt notas-aluno.txt
dougluna@dip:~/lpi/Exercicios$ wc *
     6    19    91 Script_Exemplo.sh
     6    23   123 Script_Variavel.sh
    15    27   204 alunos.txt
    93  702  4965 arquivolongo.txt
     5    10    38 codigo-aluno.txt
     5    10    21 notas-aluno.txt
    23    41   257 texto.txt
   153   832  5699 total
dougluna@dip:~/lpi/Exercicios$ |
```

Ele nos mostra as informações de cada arquivo e ao fim faz a soma de todos.

Outra forma de usar o wc é com o Pipe “ | ”. Exemplo:

```
$ cat alunos.txt | wc -l
```

Vamos ler o conteúdo do **alunos.txt** e tratar com o **wc -l**

```
dougluna@dip:~/lpi/Exercicios$ cat alunos.txt | wc -l
15
dougluna@dip:~/lpi/Exercicios$ |
```

Outra forma de usar o wc é com o Pipe “ | ”. Exemplo:

```
$ tail -n5 alunos.txt | wc
```

```
dougluna@dip:~/lpi/Exercicios$ tail -n5 alunos.txt | wc
      5      9      75
dougluna@dip:~/lpi/Exercicios$ |
```

nl

O comando **nl**, significa **number lines** é bem semelhante ao **cat -b**, ele basicamente lê o arquivo e nos mostra numerando todas as linhas que não estão em branca.

Sintaxe:

```
$ nl <ARQUIVO>
```

```
$ nl alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ nl alunos.txt
 1 Andre Gonçalves
 2 Paulo Freitas
 3 Maria Antonieto Sousa

 4 Carlos Augusto
 5 Ana Claudia
 6 Ana Claudia Vasconcelos Ana

 7 Rafael dos Santos
 8 Silvia Oliveira
 9 Antonio           Silva
10 Eliseu            Padilha
11 Ricardo
12 Douglas Luna

dougluna@dip:~/lpi/Exercicios$
```

Podemos observar que as linhas em branco foram desconsideradas.

sort

O comando **sort**, serve para ordenar alfabeticamente o conteúdo de um arquivo de texto.

Sintaxe:

```
$ sort <ARQUIVO>
```

```
$ sort alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ cat alunos.txt
Andre Gonçalves
Paulo Freitas
Maria Antonieto Sousa
```

```
Carlos Augusto
Ana Claudia
Ana Claudia Vasconcelos Ana
```

```
Rafael dos Santos
Silvia Oliveira
Antonio Silva
Eliseu Padilha
Ricardo
Douglas Luna
dougluna@dip:~/lpi/Exercicios$ sort alunos.txt
```

Sem sort

```
Ana Claudia
Ana Claudia Vasconcelos Ana
Andre Gonçalves
```

```
Antonio Silva
Carlos Augusto
Douglas Luna
Eliseu Padilha
Maria Antonieto Sousa
Paulo Freitas
Rafael dos Santos
Ricardo
Silvia Oliveira
dougluna@dip:~/lpi/Exercicios$ |
```

Com sort

```
$ sort -r <ARQUIVO>
```

-r → A opção -r (**reverse**) ordena o arquivo ao contrário:

```
dougluna@dip:~/lpi/Exercicios$ sort -r alunos.txt
Silvia Oliveira
Ricardo
Rafael dos Santos
Paulo Freitas
Maria Antonieto Sousa
Eliseu      Padilha
Douglas Luna
Carlos Augusto
Antonio      Silva
Andre Gonçalves
Ana Claudia Vasconcelos Ana
Ana Claudia
```

```
dougluna@dip:~/lpi/Exercicios$ |
```

```
$ sort -k<NUMERO> <ARQUIVO>
```

-k<NUMERO> → A opção -k<NUMERO> ordena pelo número do campo escolhido.

Vamos ordenar pelo segundo campo:

```
dougluna@dip:~/lpi/Exercicios$ sort -k2 alunos.txt

Ricardo
Antonio      Silva
Eliseu      Padilha
Maria Antonieto Sousa
Carlos Augusto
Ana Claudia
Ana Claudia Vasconcelos Ana
Paulo Freitas
Andre Gonçalves
Douglas Luna
Silvia Oliveira
Rafael dos Santos
dougluna@dip:~/lpi/Exercicios$
```

uniq

O comando **uniq**, serve para mostrar ou omitir linhas dentro do arquivo, pode mostrar as repetidas ou omiti-las. Por exemplo temos o arquivo *aluno2.txt* que o primeiro nome de alunos e em alguns casos ele está repetido. Então podemos usar o **uniq** para ver quais são as ocorrências repetidas ou quais são as únicas.

Sintaxe:

```
$ uniq <ARQUIVO>
```

```
$ uniq alunos2.txt
```

```
dougluna@dip:~/lpi/Exercicios$ cat alunos2.txt
Andre
Paulo
Maria
```

```
Carlos
Ana
Ana
```

Conteúdo original

```
Rafael
Paulo
Silvia
Antonio
Eliseu
Ricardo
Ricardo
```

```
dougluna@dip:~/lpi/Exercicios$ uniq alunos2.txt
```

```
Andre
Paulo
Maria
```

```
Carlos
Ana
```

Com comando
uniq

```
Rafael
Paulo
Silvia
Antonio
Eliseu
Ricardo
```

```
dougluna@dip:~/lpi/Exercicios$ |
```

O **uniq** tem um problema, podemos perceber que o Paulo está repetido no arquivo mesmo após executarmos o comando. Acontece que o **uniq** só entende a repetição se o conteúdo repetido estiver em sequência, uma linha após a outra, como no caso o nome “Ana” que após execução do comando ficou único, então se o nome repetido está separado o **uniq** não entende como duplicado.

Para contornar podemos seguir os passos:

1 – Ordenar o arquivo, usamos o comando sort para isso:

```
dougluna@dip:~/lpi/Exercicios$ sort alunos2.txt

Ana
Ana
Andre
Antonio
Carlos
Eliseu
Maria
Paulo
Paulo
Rafael
Ricardo
Ricardo
Silvia
dougluna@dip:~/lpi/Exercicios$ |
```

2 – podemos pegar a saída do sort, usar o Pipe “ | “ e colocarmos o **uniq**, assim ele vai trabalhar nessa saída.

```
$ sort alunos2.txt | uniq
```

```
dougluna@dip:~/lpi/Exercicios$ sort alunos2.txt | uniq

Ana
Andre
Antonio
Carlos
Eliseu
Maria
Paulo
Rafael
Ricardo
Silvia
dougluna@dip:~/lpi/Exercicios$ |
```

Agora sim ele identificou e removeu a linha duplicada.

```
$ uniq -d
```

-d → A opção **-d** mostra apenas o que está duplicado. Vamos seguir a dica anterior e usar o sort, pipe contralar a saída com uniq **-d**.

```
$ sort alunos2.txt | uniq -d
```

```
dougluna@dip:~/lpi/Exercicios$ sort alunos2.txt | uniq -d  
Ana  
Paulo  
Ricardo  
dougluna@dip:~/lpi/Exercicios$ |
```

```
$ uniq -c
```

-C → A opção **-c** conta as repetições e nos mostra o valor. Vamos seguir a dica anterior e usar o sort, pipe contralar a saída com uniq **-c**.

```
$ sort alunos2.txt | uniq -c
```

```
dougluna@dip:~/lpi/Exercicios$ sort alunos2.txt | uniq -c  
3  
2 Ana  
1 Andre  
1 Antonio  
1 Carlos  
1 Eliseu  
1 Maria  
2 Paulo  
1 Rafael  
2 Ricardo  
1 Silvia  
dougluna@dip:~/lpi/Exercicios$ |
```

Devido a característica do uniq entender apenas quando as linhas repetidas estão em sequência, ele é constantemente utilizado em conjunto com o comando sort. Então ordenamos com o sort e depois usamos o uniq.

Od

O comando od, significa octal dump e serve para exibir o conteúdo do arquivo em formato octal como padrão e outros formatos como decimal, hexadecimal etc.

Sintaxe:

```
$ od <ARQUIVO>
```

```
$ od alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ od alunos.txt
00000000 067101 071144 020145 067507 141556 060647 073154 071545
00000020 050012 072541 067554 043040 062562 072151 071541 046412
00000040 071141 060551 040440 072156 067157 062551 067564 051440
00000060 072557 060563 005012 060503 066162 071557 040440 063565
00000100 071565 067564 040412 060556 041440 060554 062165 060551
00000120 040412 060556 041440 060554 062165 060551 053040 071541
00000140 067543 061556 066145 071557 040440 060556 005012 051012
00000160 063141 062541 020154 067544 020163 060523 072156 071557
00000200 051412 066151 064566 020141 066117 073151 064545 060562
00000220 040412 072156 067157 067551 004411 064523 073154 005141
00000240 066105 071551 072545 020040 020040 020040 020040 050040
00000260 062141 066151 060550 051012 061551 071141 067544 042012
00000300 072557 066147 071541 046040 067165 005141
0000314
dougluna@dip:~/lpi/Exercicios$
```

Ele também pode ser utilizado para exibir em diversos outros formatos, vamos ver as opções:

```
Traditional format specifications may be intermixed; they accumulate:
-a same as -t a, select named characters, ignoring high-order bit
-b same as -t o1, select octal bytes
-c same as -t c, select printable characters or backslash escapes
-d same as -t u2, select unsigned decimal 2-byte units
-f same as -t fF, select floats
-i same as -t dI, select decimal ints
-l same as -t dL, select decimal longs
-o same as -t o2, select octal 2-byte units
-s same as -t d2, select decimal 2-byte units
-x same as -t x2, select hexadecimal 2-byte units
```

```
od --help
```

```
$ od -tx
```

-tx → A opção **-tx** nos mostra o conteúdo convertido para o formato hexadecimal.

```
dougluna@dip:~/lpi/Exercicios$ od -tx alunos.txt
0000000 72646e41 6f472065 61a7c36e 7365766c
0000020 7561500a 46206f6c 74696572 4d0a7361
0000040 61697261 746e4120 65696e6f 53206f74
0000060 6173756f 61430a0a 736f6c72 67754120
0000100 6f747375 616e410a 616c4320 61696475
0000120 616e410a 616c4320 61696475 73615620
0000140 636e6f63 736f6c65 616e4120 520a0a0a
0000160 65616661 6f64206c 61532073 736f746e
0000200 6c69530a 20616976 76696c4f 61726965
0000220 746e410a 6f696e6f 69530909 0a61766c
0000240 73696c45 20207565 20202020 50202020
0000260 6c696461 520a6168 72616369 440a6f64
0000300 6c67756f 4c207361 0a616e75
0000314
dougluna@dip:~/lpi/Exercicios$
```

```
$ od -t dl
```

-t dl → A opção **-t dl** nos mostra o conteúdo convertido para o formato decimal inteiros.

```
dougluna@dip:~/lpi/Exercicios$ od -t dl alunos.txt
0000000 1919184449 1866932325 1638384494 1936029292
0000020 1969311754 1176530796 1953064306 1292530529
0000040 1634300513 1953382688 1701408367 1394634612
0000060 1634956655 1631783434 1936682098 1735737632
0000100 1869902709 1634615562 1634485024 1634296949
0000120 1634615562 1634485024 1634296949 1935758880
0000140 1668181859 1936682085 1634615584 1376389642
0000160 1700882017 1868832876 1632837747 1936684142
0000200 1818841866 543254902 1986620495 1634888037
0000220 1953382666 1869180527 1767049481 174159468
0000240 1936288837 538998117 538976288 1344282656
0000260 1818846305 1376412008 1918985065 1141534564
0000300 1818719599 1277195105 174157429
0000314
dougluna@dip:~/lpi/Exercicios$ |
```

join

O comando **join** combina dois arquivos através de um índice em comum nesses arquivos. Para isso temos o arquivo código-aluno.txt que possui o número do índice e o nome do aluno, e o arquivo notas-aluno.txt que possui o número do índice e a nota do aluno.

```
dougluna@dip:~/lpi/Exercicios$ cat codigo-aluno.txt
1 Ana
2 Joao
3 Andre
4 Maria
5 Carlos
dougluna@dip:~/lpi/Exercicios$ cat notas-aluno.txt
1 10
2 8
3 9
4 2
5 0
dougluna@dip:~/lpi/Exercicios$ |
```

Sintaxe:

\$ join <ARQUIVO_1> <ARQUIVO_2>

\$ join codigo-aluno.txt notas-aluno.txt

```
dougluna@dip:~/lpi/Exercicios$ join codigo-aluno.txt notas-aluno.txt
1 Ana 10
2 Joao 8
3 Andre 9
4 Maria 2
5 Carlos 0
dougluna@dip:~/lpi/Exercicios$
```

Caso o índice seja a coluna 2 podemos usar **join** com a opção **-j2**:

```
dougluna@dip:~/lpi/Exercicios$ cat cod-prova.txt
Exame-101 1
Exame-102 2
dougluna@dip:~/lpi/Exercicios$ cat topico-prova.txt
4-topicos 1
6-topicos 2
dougluna@dip:~/lpi/Exercicios$ join -j2 cod-prova.txt topico-prova.txt
1 Exame-101 4-topicos
2 Exame-102 6-topicos
dougluna@dip:~/lpi/Exercicios$ |
```

paste

O comando **paste** combina arquivos linha a linha (merge) e não necessita de índice.

Ele basicamente pega a linha 1 do ARQUIVO_1 e a linha 1 do ARQUIVO_2, e junta os arquivos sem considerar qualquer índice ou chave.

Sintaxe:

```
$ paste <ARQUIVO_1> <ARQUIVO_2>
```

```
$ paste codigo-aluno.txt notas-aluno.txt
```

```
dougluna@dip:~/lpi/Exercicios$ paste codigo-aluno.txt notas-aluno.txt
1 Ana    1 10
2 Joao   2 8
3 Andre  3 9
4 Maria  4 2
5 Carlos      5 0
dougluna@dip:~/lpi/Exercicios$
```

Para ver mais opções use o **paste --help**

```
dougluna@dip:~/lpi/Exercicios$ paste --help
Usage: paste [OPTION]... [FILE]...
Write lines consisting of the sequentially corresponding lines from
each FILE, separated by TABs, to standard output.

With no FILE, or when FILE is -, read standard input.

Mandatory arguments to long options are mandatory for short options too.
-d, --delimiters=LIST  reuse characters from LIST instead of TABs
-s, --serial           paste one file at a time instead of in parallel
-z, --zero-terminated   line delimiter is NUL, not newline
--help                display this help and exit
--version             output version information and exit

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Report paste translation bugs to <https://translationproject.org/team/>
Full documentation at: <https://www.gnu.org/software/coreutils/paste>
or available locally via: info '(coreutils) paste invocation'
dougluna@dip:~/lpi/Exercicios$
```

split

O comando **split** divide um arquivo em vários pedaços.

Sintaxe:

```
$ split -<OPCAO> <ARQUIVO>
```

```
$ split -l20 arquivolongo.txt
$ split -20 arquivolongo.txt
```

Vamos dividir o arquivolongo.txt em outros arquivos que conterão no máximo 20.

```
dougluna@dip:~/lpi/Exercicios/split$ wc -l arquivolongo.txt
93 arquivolongo.txt
dougluna@dip:~/lpi/Exercicios/split$ split -l20 arquivolongo.txt
dougluna@dip:~/lpi/Exercicios/split$ ls -l
total 28
-rw-r--r-- 1 dougluna dougluna 4965 Jun 13 13:27 arquivolongo.txt
-rw-r--r-- 1 dougluna dougluna 1077 Jun 13 13:34 xaa
-rw-r--r-- 1 dougluna dougluna 886 Jun 13 13:34 xab
-rw-r--r-- 1 dougluna dougluna 898 Jun 13 13:34 xac
-rw-r--r-- 1 dougluna dougluna 1431 Jun 13 13:34 xad
-rw-r--r-- 1 dougluna dougluna 673 Jun 13 13:34 xae
dougluna@dip:~/lpi/Exercicios/split$ wc -l xa*
20 xaa
20 xab
20 xac
20 xad
13 xae
93 total
```

Foram criados 5 novos arquivos, e quando as linhas desses arquivos são somadas temos o mesmo valor de linhas do arquivolongo.txt

Os arquivos criados pelo **split** vieram com nomes automáticas, vamos definir que os novos devam começar com nome: **novo_arquivo_**

```
$ split -l20 arquivolongo.txt novo_arquivo_
```

```
dougluna@dip:~/lpi/Exercicios/split$ split -l20 arquivolongo.txt novo_arquivo_
dougluna@dip:~/lpi/Exercicios/split$ ls -l
total 28
-rw-r--r-- 1 dougluna dougluna 4965 Jun 13 13:27 arquivolongo.txt
-rw-r--r-- 1 dougluna dougluna 1077 Jun 13 13:38 novo_arquivo_aa
-rw-r--r-- 1 dougluna dougluna 886 Jun 13 13:38 novo_arquivo_ab
-rw-r--r-- 1 dougluna dougluna 898 Jun 13 13:38 novo_arquivo_ac
-rw-r--r-- 1 dougluna dougluna 1431 Jun 13 13:38 novo_arquivo_ad
-rw-r--r-- 1 dougluna dougluna 673 Jun 13 13:38 novo_arquivo_ae
dougluna@dip:~/lpi/Exercicios/split$ |
```

-b<NUMERO> → Com a opção **-b<NUMERO>** podemos dividir por bytes.

```
$ split -b500 arquivolongo.txt arquivo_500bytes_
```

```
dougluna@dip:~/Exercicios/split$ split -b500 arquivolongo.txt arquivo_500bytes_
dougluna@dip:~/Exercicios/split$ ls -l
total 48
-rw-r--r-- 1 dougluna dougluna 500 Jun 13 13:42 arquivo_500bytes_aa
-rw-r--r-- 1 dougluna dougluna 500 Jun 13 13:42 arquivo_500bytes_ab
-rw-r--r-- 1 dougluna dougluna 500 Jun 13 13:42 arquivo_500bytes_ac
-rw-r--r-- 1 dougluna dougluna 500 Jun 13 13:42 arquivo_500bytes_ad
-rw-r--r-- 1 dougluna dougluna 500 Jun 13 13:42 arquivo_500bytes_ae
-rw-r--r-- 1 dougluna dougluna 500 Jun 13 13:42 arquivo_500bytes_af
-rw-r--r-- 1 dougluna dougluna 500 Jun 13 13:42 arquivo_500bytes_ag
-rw-r--r-- 1 dougluna dougluna 500 Jun 13 13:42 arquivo_500bytes_ah
-rw-r--r-- 1 dougluna dougluna 500 Jun 13 13:42 arquivo_500bytes_ai
-rw-r--r-- 1 dougluna dougluna 465 Jun 13 13:42 arquivo_500bytes_aj
-rw-r--r-- 1 dougluna dougluna 4965 Jun 13 13:27 arquivolongo.txt
dougluna@dip:~/Exercicios/split$ wc -c arquivolongo.txt
4965 arquivolongo.txt
dougluna@dip:~/Exercicios/split$ wc -c arquivo_500bytes_a*
500 arquivo_500bytes_aa
500 arquivo_500bytes_ab
500 arquivo_500bytes_ac
500 arquivo_500bytes_ad
500 arquivo_500bytes_ae
500 arquivo_500bytes_af
500 arquivo_500bytes_ag
500 arquivo_500bytes_ah
500 arquivo_500bytes_ai
465 arquivo_500bytes_aj
4965 total
dougluna@dip:~/Exercicios/split$ |
```

Foram criados arquivos com 500 bytes cada e soma deles dão o mesmo valor que o arquivolongo.txt que foi a origem da divisão.

tr

O comando **tr**, vem da palavra “translate” e tem como principais funções substituir ou apagar caracteres conforme sua escolha. Ele é diferente dos demais comandos, pois controlamos seu uso diretamente com a ajuda do Pipe “|”.

```
dougluna@dip:~/Exercicios$ tr --help
Usage: tr [OPTION]... SET1 [SET2]
Translate, squeeze, and/or delete characters from standard input,
writing to standard output.

-c, -C, --complement      use the complement of SET1
-d, --delete                delete characters in SET1, do not translate
-s, --squeeze-repeats     replace each sequence of a repeated character
                           that is listed in the last specified SET,
                           with a single occurrence of that character
-t, --truncate-set1       first truncate SET1 to length of SET2
--help          display this help and exit
--version        output version information and exit
```

Exemplo:

```
$ cat alunos.txt | tr a-z A-Z
```

Vamos substituir todos os caracteres minúsculos de a-z por caracteres maiúsculos A-Z.

```
dougluna@dip:~/lpi/Exercicios$ cat alunos.txt | tr a-z A-Z
ANDRE GONÇALVES
PAULO FREITAS
MARIA ANTONIETO SOUSA

CARLOS AUGUSTO
ANA CLAUDIA
ANA CLAUDIA VASCONCELOS ANA

RAFAEL DOS SANTOS
SILVIA OLIVEIRA
ANTONIO      SILVA
ELISEU      PADILHA
RICARDO
DOUGLAS LUNA
```

```
$ cat alunos.txt | tr [:lower:] [:upper:]
```

Podemos usar as opções específicas do **tr** para substituir todos os caracteres minúsculos [:lower:] por caracteres maiúsculos [:upper:].

```
SETS are specified as strings of characters. Most represent themselves.
Interpreted sequences are:
```

\NNN	character with octal value NNN (1 to 3 octal digits)
\\	backslash
\a	audible BEL
\b	backspace
\f	form feed
\n	new line
\r	return
\t	horizontal tab
\v	vertical tab
CHAR1-CHAR2	all characters from CHAR1 to CHAR2 in ascending order
[CHAR*]	in SET2, copies of CHAR until length of SET1
[CHAR*REPEAT]	REPEAT copies of CHAR, REPEAT octal if starting with 0
[:alnum:]	all letters and digits
[:alpha:]	all letters
[:blank:]	all horizontal whitespace
[:cntrl:]	all control characters
[:digit:]	all digits
[:graph:]	all printable characters, not including space
[:lower:]	all lower case letters
[:print:]	all printable characters, including space
[:punct:]	all punctuation characters
[:space:]	all horizontal or vertical whitespace
[:upper:]	all upper case letters
[:xdigit:]	all hexadecimal digits
[=CHAR=]	all characters which are equivalent to CHAR

```
dougluna@dip:~/lpi/Exercicios$ cat alunos.txt | tr [:lower:] [:upper:]
ANDRE GONÇALVES
PAULO FREITAS
MARIA ANTONIETO SOUSA

CARLOS AUGUSTO
ANA CLAUDIA
ANA CLAUDIA VASCONCELOS ANA

RAFAEL DOS SANTOS
SILVIA OLIVEIRA
ANTONIO      SILVA
ELISEU       PADILHA
RICARDO
DOUGLAS LUNA
```

```
$ cat alunos.txt | tr " " "-"
```

Também é possível substituir caracteres específicos, aqui vamos substituir todos os espaços pelo hífen.

```
dougluna@dip:~/lpi/Exercicios$ cat alunos.txt | tr " " "-"
Andre-Gonçalves
Paulo-Freitas
Maria-Antonieto-Sousa

Carlos-Augusto
Ana-Claudia
Ana-Claudia-Vasconcelos-Ana

Rafael-dos-Santos
Silvia-Oliveira
Antonio           Silva
Eliseu-----Padilha
Ricardo
Douglas-Luna
```

-d → Delete, podemos usar a opção do **tr -d** para deletar caracteres escolhidos.
Exemplo para deletar todos os caracteres maiúsculos [:upper:].

```
$ cat alunos.txt | tr -d [:upper:]
```

```
dougluna@dip:~/lpi/Exercicios$ cat alunos.txt | tr -d [:upper:]
ndre onçalves
aulo reitas
aria ntonieto ousa

arlos ugusto
na laudia
na laudia asconcelos na

afael dos antos
ilvia liveira
ntonio           ilva
liseu            adilha
icardo
ougglas una
```

Comando foi executado com sucesso. Para ver mais opções do **tr** use o **tr --help**.

-s, --squeeze-repeats → Substitui a sequência de caracteres repetidos que foi indicado, por apenas uma ocorrência desse caractere.

```
$ echo "Curso de Liiinux" | tr -s i
```

Aqui vamos indicar que o caractere “ i ” está repetindo e queremos substituir essa sequência por apenas um “ i ” .

```
dougluna@dip:~/lpi/Exercicios$ echo "Curso de Liiiiinux!"  
Curso de Liiiiinux!  
dougluna@dip:~/lpi/Exercicios$ echo "Curso de Liiiiinux!" | tr -s i  
Curso de Linux!  
dougluna@dip:~/lpi/Exercicios$ |
```

Outro uso comum é para tirar os espaços repetidos.

```
$ echo "Curso de Liiinux" | tr -s " "
```

```
dougluna@dip:~/lpi/Exercicios$ echo "Curso de Linux!"  
Curso de Linux!  
dougluna@dip:~/lpi/Exercicios$ echo "Curso de Linux!" | tr -s " "  
Curso de Linux!  
dougluna@dip:~/lpi/Exercicios$ |
```

Um caso usual de aplicação do **tr** são na conversão de arquivo de textos vindos do sistema operacional Windows.

No Linux é usado para a quebra de linha:

LF – Representado por: \$

LF = Line Feed = Nova Linha

\n

No Windows é usado para a quebra de linha:

CR-LF – Representado por: ^M

CR = Carriage Return

\r

LF = Line Feed = Nova Linha

\n

Usamos o **cat -A <arquivo>** para ver todo o conteúdo do arquivo.

Usamos o **file <arquivo>** para ver o tipo do arquivo.

```
dougluna@dip:~$ cat -A ArquivoGeradoLinux.txt
Este arquivo foi gerado em$
um vi simples do Linux$
$
para exemplificar a aula sobre o comando tr.$
$
dougluna@dip:~$ cat -A ArquivoGeradoWindows.txt
Este arquivo foi gerado em^M$
um notepad simples do Windows^M$
^M$
para exemplificar a aula sobre o comando tr.^M$
dougluna@dip:~$ file ArquivoGerado*
ArquivoGeradoLinux.txt: ASCII text
ArquivoGeradoWindows.txt: ASCII text, with CRLF line terminators
dougluna@dip:~$ |
```

Comparando os arquivos com o comando **od -c** podemos ver as diferenças também, onde o arquivo gerado no Windows tem o \r.

```
dougluna@dip:~$ od -c ArquivoGeradoLinux.txt
0000000 E s t e a r q u i v o f o r m a t o d e l i n h a s
00000020 g e r a d o e m \n u m v i
00000040 s i m p l e s d o L i n u x
00000060 x \n \n p a r a e x e m p l i f i c a r a u l a s o b r e o c o m a n d o t r .
00000100 i c a r a a u l a s o b r e o c o m a n d o t r .
00000120 e o c o m a n d o t r . \n
00000140 \n
00000141
dougluna@dip:~$ od -c ArquivoGeradoWindows.txt
0000000 E s t e a r q u i v o f o r m a t o d e l i n h a s
00000020 g e r a d o e m \r \n u m v i
00000040 o t e p a d o s i m p l e s d e l i n h a s
00000060 o W i n d o w s \r \n \r \n p a r a e x e m p l i f i c a r a u l a s o b r e o c o m a n d o t r .
00000100 a e x e m p l i f i c a r a u l a s o b r e o c o m a n d o t r .
00000120 a u l a s o b r e o c o m a n d o t r . \r \n
00000140 m a n d o t r . \r \n
00000153
```

Vamos ajustar o arquivo gerado no Windows com o **tr -d** onde deletamos o \r e redirecionamos a saída para um novo arquivo.

```
$ tr -d "\r" < ArquivoGeradoWindows.txt > NovoArquivo.txt
```

```
dougluna@dip:~$ tr -d "\r" < ArquivoGeradoWindows.txt > NovoArquivo.txt
dougluna@dip:~$ file ArquivoGerado* NovoArquivo.txt
ArquivoGeradoLinux.txt: ASCII text
ArquivoGeradoWindows.txt: ASCII text, with CRLF line terminators
NovoArquivo.txt: ASCII text
dougluna@dip:~$ |
```

Dessa forma o NovoArquivo.txt passa ser no padrão gerado no Unix/Linux.

Cut

O comando **cut** é usado para recortar seções de texto de cada linha do arquivo. Podemos delimitar um texto por bytes, caracteres ou campo, e recortar apenas a parte que nos interessa.

```
dougluna@dip:~/Exercicios$ cut --help
Usage: cut OPTION... [FILE]...
Print selected parts of lines from each FILE to standard output.

With no FILE, or when FILE is -, read standard input.

Mandatory arguments to long options are mandatory for short options too.
-b, --bytes=LIST      select only these bytes
-c, --characters=LIST select only these characters
-d, --delimiter=DELIM  use DELIM instead of TAB for field delimiter
-f, --fields=LIST      select only these fields; also print any line
                        that contains no delimiter character, unless
                        the -s option is specified
-n                   (ignored)
--complement        complement the set of selected bytes, characters
                     or fields
-s, --only-delimited do not print lines not containing delimiters
                     --output-delimiter=STRING use STRING as the output delimiter
                     the default is to use the input delimiter
-z, --zero-terminated line delimiter is NUL, not newline
--help            display this help and exit
--version         output version information and exit
```

Sintaxe:

```
$ cut -<OPCAO> <ARQUIVO>
```

```
$ cut -c1-5 alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ cut -c1-5 alunos.txt
Andre
Paulo
Maria

Carlo
Ana C
Ana C

Rafae
Silvi
Anton
Elise
Ricar
Dougl
dougluna@dip:~/lpi/Exercicios$
```

Usando a opção do **cut -c**, delimitamos a partir dos caracteres e nesse exemplo recortamos apenas do caractere 1 ao 5 **1-5** do arquivo **alunos.txt**

Podemos fazer isso de diversas formas:

```
dougluna@dip:~/lpi/Exercicios$ cut -c5-10 topico-prova.txt
picos                               caractere 5 ao 10
picos
dougluna@dip:~/lpi/Exercicios$ cut -c5- topico-prova.txt
picos 1                             caractere 5 até o final
picos 2
dougluna@dip:~/lpi/Exercicios$ cut -c-5 topico-prova.txt
4-top                                do começo até o caractere 5
6-top
dougluna@dip:~/lpi/Exercicios$ |
```

Podemos definir apenas os caracteres que desejamos:

```
$ cut -c1,3,5 alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ cut -c1,3,5 topico-prova.txt
4tp
6tp
dougluna@dip:~/lpi/Exercicios$
```

-b, --bytes → Selecionar apenas os bytes desejados.

```
$ cut -b1-5 alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ cut -b1-5 alunos.txt
Andre
Paulo
Maria

Carlo
Ana C
Ana C

Rafae
Silvi
Anton
Elise
Ricar
Dougl
```

Recortamos com o comando **cut** e a opção **-b** do byte 1 até o 5 **1-5**

-f, --fields → Essa funcionalidade é bastante usada para selecionar apenas os campos desejados, e junto dele definimos qual é delimitador dos campos, como um espaço ou uma vírgula.

-d, --delimiter → Aqui usamos a opção **-d“ ”** para definir que o delimitador é um espaço, e o **-f1** para recortarmos apenas o campo 1.

```
$ cut -d" " -f1 alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ cut -d" " -f1 alunos.txt
Andre
Paulo
Maria

Carlos
Ana
Ana

Rafael
Silvia
Antonio      Silva
Eliseu
Ricardo
Douglas
```

Da mesma forma podemos usar os ranges para pegar os campos 2 e 3 por exemplo:

```
$ cut -d" " -f2-3 topico-prova.txt
```

```
dougluna@dip:~/lpi/Exercicios$ cut -d" " -f2-3 topico-prova.txt
1
2
dougluna@dip:~/lpi/Exercicios$ cut -d" " -f2-3 topico-prova.txt
4-topicos 1
6-topicos 2
dougluna@dip:~/lpi/Exercicios$ cut -d" " -f2- topico-prova.txt
1
2
dougluna@dip:~/lpi/Exercicios$ |
```

sed

sed - stream editor, usado para filtrar e transformar texto. Geralmente é utilizado principalmente em conjunto com expressões regulares. Vamos ver algumas funcionalidades mais simples dele e depois nos aprofundaremos no uso com expressões regulares. Com o **sed** podemos procurar um conteúdo e substituir e/ou deletar uma parte do texto.

https://www.gnu.org/software/sed/manual/html_node/index.html#SEC_Contents

Sintaxe:

\$ sed -<SCRIPT> <ARQUIVO>

```
$ sed 's/Ana/Maria/' alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ cat alunos.txt
Andre Gonçalves
Paulo Freitas
Maria Antonieto Sousa

Carlos Augusto
Ana Claudia
Ana Claudia Vasconcelos Ana

Rafael dos Santos
Silvia Oliveira
Antonio      Silva
Eliseu       Padilha
Ricardo
Douglas Luna
dougluna@dip:~/lpi/Exercicios$ sed 's/Ana/Maria/' alunos.txt
Andre Gonçalves
Paulo Freitas
Maria Antonieto Sousa

Carlos Augusto
Maria Claudia
Maria Claudia Vasconcelos Ana

Rafael dos Santos
Silvia Oliveira
Antonio      Silva
Eliseu       Padilha
Ricardo
Douglas Luna
```

Ainda temos o nome “Ana” aparecendo, isso acontece pois ele analisa linha por linha e apenas a primeira ocorrência da linha que ele encontrou é substituída.

```
dougluna@dip:~/Exercicios$ sed --help
Usage: sed [OPTION]... {script-only-if-no-other-script} [input-file]...

-n, --quiet, --silent
    suppress automatic printing of pattern space
--debug
    annotate program execution
-e script, --expression=script
    add the script to the commands to be executed
-f script-file, --file=script-file
    add the contents of script-file to the commands to be executed
--follow-symlinks
    follow symlinks when processing in place
-i[SUFFIX], --in-place[=SUFFIX]
    edit files in place (makes backup if SUFFIX supplied)
-l N, --line-length=N
    specify the desired line-wrap length for the 'l' command
--posix
    disable all GNU extensions.
-E, -r, --regexp-extended
    use extended regular expressions in the script
    (for portability use POSIX -E).
-s, --separate
    consider files as separate rather than as a single,
    continuous long stream.
--sandbox
    operate in sandbox mode (disable e/r/w commands).
-u, --unbuffered
    load minimal amounts of data from the input files and flush
    the output buffers more often
-z, --null-data
    separate lines by NUL characters
--help    display this help and exit
--version output version information and exit
```

Para que seja substituído tudo, devemos usar uma **flag**, no final do comando que é o **/g faz que a substituição seja GLOBAL**.

```
$ sed 's/Ana/Maria/g' alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ sed 's/Ana/Maria/g' alunos.txt
Andre Gonçalves
Paulo Freitas
Maria Antonieto Sousa

Carlos Augusto
Maria Claudia
Maria Claudia Vasconcelos Maria

Rafael dos Santos
Silvia Oliveira
Antonio      Silva
Eliseu       Padilha
Ricardo
Douglas Luna
```

Agora podemos notar que a linha onde a palavra Ana se repetia, todos foram substituídos por Maria.

'd' → Usamos para deletar algum texto do arquivo. No exemplo abaixo vamos deletar das linhas 3 até a 5.

```
$ sed '3,5 d' alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ sed '3,5 d' alunos.txt
Andre Gonçalves
Paulo Freitas
Ana Claudia
Ana Claudia Vasconcelos Ana

Rafael dos Santos
Silvia Oliveira
Antonio      Silva
Eliseu       Padilha
Ricardo
Douglas Luna
```

Outra forma de deletar é buscando a palavra junto ao 'd' e toda vez que ele encontrar essa palavra o sed **vai remover a linha toda** onde a mesma se encontra

```
$ sed '/Claudia/d' alunos.txt
```

```
dougluna@dip:~/lpi/Exercicios$ cat -n alunos.txt
 1 Andre Gonçalves
 2 Paulo Freitas
 3 Maria Antonieto Sousa
 4
 5 Carlos Augusto
 6 Ana Claudia
 7 Ana Claudia Vasconcelos Ana
 8
 9
10 Rafael dos Santos
11 Silvia Oliveira
12 Antonio           Silva
13 Eliseu            Padilha
14 Ricardo
15 Douglas Luna
dougluna@dip:~/lpi/Exercicios$ sed '/Claudia/d' alunos.txt
Andre Gonçalves
Paulo Freitas
Maria Antonieto Sousa
Carlos Augusto
Rafael dos Santos
Silvia Oliveira
Antonio           Silva
Eliseu            Padilha
Ricardo
Douglas Luna
```

As linhas com a palavra Claudia foram excluídas

xzcat

xzcat, nos permite visualizar o conteúdo de um arquivo compactado com o algoritmo XZ compressed data (.xz e .lzma)

Para compactar um arquivo com esse algoritmo use o comando **xz**.

Sintaxe:

```
$ xzcat <ARQUIVO>.xz
```

```
$ xzcat arquivolongo.txt.xz
```

```
dougluna@dip:~/lpi/Exercicios$ file arquivolongo.txt.xz
arquivolongo.txt.xz: XZ compressed data
dougluna@dip:~/lpi/Exercicios$ xzcat arquivolongo.txt.xz
exemplo do ZZ

LPI Linux Essentials
ooooasfafja
aoaoaoa
aaaa
```

```
dougluna@dip:~/lpi/Exercicios$ xz --help
Usage: xz [OPTION]... [FILE]...
Compress or decompress FILEs in the .xz format.

-z, --compress      force compression
-d, --decompress   force decompression
-t, --test         test compressed file integrity
-l, --list          list information about .xz files
-k, --keep          keep (don't delete) input files
-f, --force         force overwrite of output file and (de)compress links
-c, --stdout        write to standard output and don't delete input files
-@ ... -9           compression preset; default is 6; take compressor *and*
                    decompressor memory usage into account before using 7-9!
-e, --extreme       try to improve compression ratio by using more CPU time;
                    does not affect decompressor memory requirements
-T, --threads=NUM   use at most NUM threads; the default is 1; set to 0
                    to use as many threads as there are processor cores
-q, --quiet         suppress warnings; specify twice to suppress errors too
-v, --verbose       be verbose; specify twice for even more verbose
-h, --help          display this short help and exit
-H, --long-help     display the long help (lists also the advanced options)
-V, --version       display the version number and exit

With no FILE, or when FILE is -, read standard input.

Report bugs to <lasse.collin@tukaani.org> (in English or Finnish).
XZ Utils home page: <https://tukaani.org/xz/>
```

bzcat

O comando **bzcat**, nos permite visualizar o conteúdo de um arquivo compactado com o algoritmo **bzip2 compressed data**.

Para compactar um arquivo com esse algoritmo use o comando **bzip2**.

Sintaxe:

```
$ bzcat <ARQUIVO>.bz2
```

```
$ bzcat arquivolongo.txt.bz2
```

```
dougluna@dip:~/lpi/Exercicios$ file arquivolongo.txt.bz2
arquivolongo.txt.bz2: bzip2 compressed data, block size = 900k
dougluna@dip:~/lpi/Exercicios$ bzcat arquivolongo.txt.bz2
exemplo do ZZ
```

```
LPI Linux Essentials
ooooasfafja
aoaoaoa
aaaa
```

```
dougluna@dip:~/lpi/Exercicios$ bzip2 --help
bzip2, a block-sorting file compressor. Version 1.0.8, 13-Jul-2019.

usage: bzip2 [flags and input files in any order]

-h --help      print this message
-d --decompress  force decompression
-z --compress   force compression
-k --keep       keep (don't delete) input files
-f --force      overwrite existing output files
-t --test       test compressed file integrity
-c --stdout     output to standard out
-q --quiet      suppress noncritical error messages
-v --verbose    be verbose (a 2nd -v gives more)
-L --license    display software version & license
-V --version    display software version & license
-s --small      use less memory (at most 2500k)
-1 .. -9        set block size to 100k .. 900k
--fast          alias for -1
--best          alias for -9

If invoked as 'bzip2', default action is to compress.
as 'bunzip2', default action is to decompress.
as 'bzcat', default action is to decompress to stdout.

If no file names are given, bzip2 compresses or decompresses
from standard input to standard output. You can combine
short flags, so '-v -4' means the same as -v4 or -4v, &c.
```

bzip2 --help

Zcat

gzcat - nos permite visualizar o conteúdo de um arquivo compactado com o algoritmo **gzip compressed data**.

Para compactar um arquivo com esse algoritmo use o comando **gzip**.

Sintaxe:

```
$ zcat <ARQUIVO>.gz
```

```
$ zcat arquivolongo.txt.gz
```

```
dougluna@dip:~/lpi/Exercicios$ file arquivolongo.txt.gz
arquivolongo.txt.gz: gzip compressed data, was "arquivolongo.txt", last modified: Thu May 18 14:23:02 2017, from Unix, o
riginal size modulo 2^32 4965
dougluna@dip:~/lpi/Exercicios$ zcat arquivolongo.txt.gz
exemplo do ZZ

LPI Linux Essentials
ooooasfafja
aoaoaoa
aaaa
```

```
dougluna@dip:~/lpi/Exercicios$ gzip --help
Usage: gzip [OPTION]... [FILE]...
Compress or uncompress FILEs (by default, compress FILES in-place).

Mandatory arguments to long options are mandatory for short options too.

-c, --stdout      write on standard output, keep original files unchanged
-d, --decompress  decompress
-f, --force        force overwrite of output file and compress links
-h, --help         give this help
-k, --keep        keep (don't delete) input files
-l, --list         list compressed file contents
-L, --license     display software license
-n, --no-name     do not save or restore the original name and timestamp
-N, --name         save or restore the original name and timestamp
-q, --quiet        suppress all warnings
-r, --recursive   operate recursively on directories
      --rsyncable  make rsync-friendly archive
-S, --suffix=SUF  use suffix SUF on compressed files
      --synchronous synchronous output (safer if system crashes, but slower)
-t, --test         test compressed file integrity
-v, --verbose     verbose mode
-V, --version     display version number
-1, --fast        compress faster
-9, --best        compress better

With no FILE, or when FILE is -, read standard input.

Report bugs to <bug-gzip@gnu.org>.
```

Comandos de checksum

Os comandos de checksum são usados para detectar corrupção accidental ou proposital de arquivos. Ele é muito usado na validação de imagens .iso de sistemas operacional, onde o fornecedor lhe dá a hash do checksums e você pode validar após o download da .iso se o checksum é o mesmo, dessa forma a sua imagem .iso pode ser considerada intacta , sem nenhuma byte a mais ou a menos.

md5sum

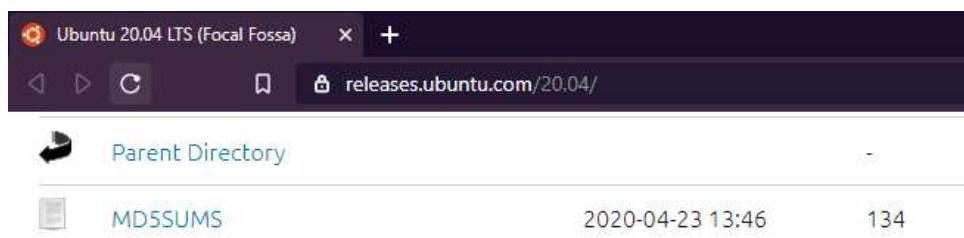
Calcula um checksum (soma de verificação) de 128 bits (ou "fingerprint" ou message-digest") para cada arquivo especificado. O resumo MD5 é mais confiável que um simples CRC (fornecido pelo comando 'cksum') para detectar corrupção accidental de arquivos, como o chances de ter accidentalmente dois arquivos com MD5 idêntico são incrivelmente pequenos.

```
$ md5sum ubuntu-20.04-desktop-amd64.iso
```

```
dougluna@dip:/mnt/d/Documents/Iso$ md5sum ubuntu-20.04-desktop-amd64.iso
ea28c4fd933be55f9f01a5fa9e868490  ubuntu-20.04-desktop-amd64.iso
dougluna@dip:/mnt/d/Documents/Iso$ |
```

Podemos fazer o download do arquivo MD5SUMS no site do dono da distribuição e realizarmos a comparação.

```
$ md5sum -c MD5SUMS
```



```
dougluna@dip:/mnt/d/Documents/Iso$ ls
MD5SUMS  Windows.iso  ubuntu-20.04-desktop-amd64.iso
dougluna@dip:/mnt/d/Documents/Iso$ md5sum -c MD5SUMS
ubuntu-20.04-desktop-amd64.iso: OK ←
md5sum: ubuntu-20.04-live-server-amd64.iso: No such file or directory
ubuntu-20.04-live-server-amd64.iso: FAILED open or read
md5sum: WARNING: 1 listed file could not be read
```

Obs: o arquivo dos SUMS e a isso precisam estar no mesmo diretório.

sha256sum

Calcula um checksum (soma de verificação) de 256 bits (ou "fingerprint" ou "message-digest") para cada arquivo especificado. O resumo SHA256SUM é mais confiável que o md5sum para detectar corrupção accidental de arquivos.

```
$ sha256sum ubuntu-20.04-desktop-amd64.iso
```

```
dougluna@dip:~/Exercicios$ sha256sum texto.txt
95a9e4c2516fadbadbd7f8d17667518c2ad14cfce39d9f7f78f7faaa47086a34  texto.txt
```

	File Name	File Size	Date
Parent directory/	-	-	-
FOOTER.html	810 B	2020-Aug-06 15:22	
HEADER.html	4.1 KIB	2020-Aug-06 15:22	
SHA256SUMS	198 B	2020-Aug-06 15:22	
SHA256SUMS.gpg	833 B	2020-Aug-06 15:22	
xubuntu-20.04.1-desktop-amd64.iso	1.6 GiB	2020-Jul-31 16:51	
xubuntu-20.04.1-desktop-amd64.iso.torrent	126.9 KIB	2020-Aug-06 15:22	
xubuntu-20.04.1-desktop-amd64.iso.zsync	3.2 MiB	2020-Aug-06 15:22	
xubuntu-20.04.1-desktop-amd64.list	8.4 KIB	2020-Jul-31 16:51	
xubuntu-20.04.1-desktop-amd64.manifest	50.0 KiB	2020-Jul-31 16:43	

Podemos fazer o download do arquivo SHA256SUMS no site do dono da distribuição e realizarmos a comparação.

```
$ sha256sum -c SHA256SUMS
```

```
lpil@linux:~$ sha256sum -c SHA256SUMS
kubuntu-18.04-desktop-amd64.iso: OK
sha256sum: xubuntu-18.04-desktop-i386.iso: No such file or directory
xubuntu-18.04-desktop-i386.iso: FAILED open or read
sha256sum: xubuntu-18.04.1-desktop-amd64.iso: No such file or directory
xubuntu-18.04.1-desktop-amd64.iso: FAILED open or read
sha256sum: xubuntu-18.04.1-desktop-i386.iso: No such file or directory
xubuntu-18.04.1-desktop-i386.iso: FAILED open or read
sha256sum: WARNING: 3 listed files could not be read
lpil@linux:~$
```

sha512sum

Calcula um checksum (soma de verificação) de 512 bits (ou "fingerprint" ou message-digest") para cada arquivo especificado. O SHA512SUM é mais confiável que o sha256sum para detectar corrupção accidental de arquivos.

```
$ sha512sum ubuntu-20.04-desktop-amd64.iso
```

```
dougluna@dip:~/Exercicios$ sha512sum texto.txt
3d80c8debe27bd6d8a6e0a6644197a6d95edb42aa7e2fa69f05f9127471348ce733f04ab89e3c0381503406ea59b3daeb8a69ff927f5f86dd0e9d7c0d50ad808  texto.txt
```

Podemos fazer o download do arquivo SHA512SUMS no site do dono da distribuição e realizarmos a comparação.

```
$ sha512sum -c SHA512SUMS
```