103.2 Processar fluxos de texto usando filtros

Referência ao LPI objectivo

LPIC-1 v5, Exam 101, Objective 103.2

Peso

2

Áreas chave de conhecimento

- Enviar arquivos de texto e saídas de fluxo de textos através de filtros para modificar a saída usando comandos padrão UNIX encontrados no pacote GNU textutils.

Segue uma lista parcial dos arquivos, termos e utilitários utilizados

- `bzcat`
- `cat`
- `cut`
- `head`
- `less`
- `md5sum`
- `nl`
- `od`
- `paste`
- `sed`
- `sha256sum`
- `sha512sum`

- `sort`
- `split`
- `tail`
- `tr`
- `uniq`
- `wc`
- `xzcat`
- `zcat`

103.2 Lição 1

Certificação:	LPIC-1
Versão:	5.0
Tópico:	103 Comandos GNU e Unix
Objetivo:	103.2 Processar fluxos de texto usando filtros
Lição:	1 de 1

Introdução

Lidar com textos é uma parte importante do trabalho de todo administrador de sistemas. Doug McIlroy, membro da equipe de desenvolvimento original do Unix, resumiu a filosofia Unix e disse (dentre outras coisas importantes): "Escreva programas para lidar com fluxos de texto, porque essa é uma interface universal." O Linux é inspirado no sistema operacional Unix e defende essa mesma filosofia; portanto, um administrador deve estar pronto para lidar com muitas ferramentas de manipulação de texto dentro de uma distribuição Linux.

Uma revisão rápida sobre redirecionamentos e pipes

Também da filosofia Unix:

- Escreva programas que façam apenas uma coisa e a façam bem.
- Escreva programas que trabalhem juntos.

Uma das principais formas de fazer os programas trabalharem juntos é usar *piping* (canalização) e *redirecionamentos*. Praticamente todos os programas de manipulação de texto obtêm texto de uma entrada padrão (*stdin*), produzem uma saída padrão (*stdout*) e enviam eventuais erros para uma saída

de erro padrão (*stderr*). A menos que especifiquemos o contrário, a entrada padrão será o que digitamos no teclado (o programa lê esse texto quando pressionamos Enter). Da mesma forma, a saída padrão e os erros serão exibidos na tela do terminal. Vamos ver como isso funciona.

Em seu terminal, digite `cat` e pressione a tecla Enter. Em seguida, digite algum texto aleatório.

```
$ cat
This is a test
This is a test
Hey!
Hey!
It is repeating everything I type!
It is repeating everything I type!
(I will hit ctrl+c so I will stop this nonsense)
(I will hit ctrl+c so I will stop this nonsense)
^C
```

Para saber mais sobre o comando `cat` (o termo vem de "concatenar"), consulte as páginas de manual.

NOTE

Se estiver trabalhando em uma instalação simples de um servidor Linux, alguns comandos como `info` e `less` talvez não estejam disponíveis. Se for este o caso, instale essas ferramentas usando o procedimento adequado em seu sistema, conforme descrito nas lições correspondentes.

Conforme demonstrado acima, se você não especificar de onde `cat` deve ler, ele lerá a entrada padrão (o que você digitar) e produzirá o que for lido na janela do terminal (a saída padrão).

Agora tente o seguinte:

```
$ cat > mytextfile
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
^C

$ cat mytextfile
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
```

O sinal de `>` (maior que) diz ao `cat` para enviar a saída ao arquivo `mytextfile`, não para a saída

padrão. Agora tente o seguinte:

```
$ cat mytextfile > mynewtextfile
$ cat mynewtextfile
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
```

O efeito desse comando é copiar `mytextfile` para `mynewtextfile`. Na verdade, podemos verificar se esses dois arquivos têm o mesmo conteúdo executando um `diff`:

```
$ diff mynewtextfile mytextfile
```

Como não há saída, os arquivos são idênticos. Agora, tente o operador de redirecionamento de acréscimo (`>>`):

```
$ echo 'This is my new line' >> mynewtextfile
$ diff mynewtextfile mytextfile
4d3
< This is my new line
```

Até agora, usamos redirecionamentos para criar e manipular arquivos. Também podemos usar pipes (representados pelo símbolo `|`) para redirecionar a saída de um programa para outro programa. Vamos encontrar as linhas onde a palavra “this” aparece:

```
$ cat mytextfile | grep this
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this

$ cat mytextfile | grep -i this
This is a test
I hope cat is storing this to mytextfile as I redirected the output
I will hit ctrl+c now and check this
```

Nós canalizamos a saída de `cat` para outro comando: `grep`. Observe que, quando ignoramos o uso de maiúsculas e minúsculas (usando a opção `-i`), obtemos uma linha extra como resultado.

Processando fluxos de texto

Lendo um arquivo compactado

Vamos criar um arquivo chamado `ftu.txt` contendo uma lista dos seguintes comandos:

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

A seguir, vamos usar o comando `grep` para imprimir todas as linhas que contêm a string `cat`:

```
$ cat ftu.txt | grep cat
bzcat
cat
xzcat
zcat
```

Outra forma de obter essas informações é apenas usar o comando `grep` para filtrar o texto diretamente, sem a necessidade de usar outro aplicativo para enviar o fluxo de texto para `stdout`.

```
$ grep cat ftu.txt
bzcat
cat
xzcat
```

```
zcat
```

NOTE Lembre-se de que há muitas maneiras de realizar uma tarefa usando o Linux.

Existem outros comandos que lidam com arquivos compactados (`bzcat` para arquivos compactados `bzip`, `xzcat` para arquivos `xz` e `zcat` para arquivos `gzip`). Eles são usados para visualizar o conteúdo de um arquivo compactado com base no algoritmo de compactação empregado.

Verifique se o arquivo recém-criado `ftu.txt` é o único no diretório e, em seguida, crie uma versão compactada `gzip` do arquivo:

```
$ ls ftu*
ftu.txt

$ gzip ftu.txt
$ ls ftu*
ftu.txt.gz
```

Em seguida, use o comando `zcat` para visualizar o conteúdo do arquivo compactado com o gzip:

```
$ zcat ftu.txt.gz
bzcat
cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
xzcat
zcat
```

Note que `gzip` comprime `ftu.txt` em `ftu.txt.gz` e remove o arquivo original. Por padrão, nenhuma saída do comando `gzip` é exibida. Porém, se você deseja que o `gzip` lhe diga o que está fazendo, use a opção `-v` para a saída "verbosa".

Visualizando um arquivo no paginador

Sabemos que `cat` concatena um arquivo para a saída padrão (quando um arquivo é informado após o comando). O arquivo `/var/log/syslog` é onde o sistema Linux armazena tudo de importante que acontece no sistema. Usamos o comando `sudo` para elevar nossos privilégios e poder ler o arquivo `/var/log/syslog`:

```
$ sudo cat /var/log/syslog
```

...e veremos mensagens rolando muito rápido na janela do terminal. É possível canalizar a saída para o programa `less` para que os resultados sejam paginados. Com o `less`, podemos usar as setinhas do teclado para navegar pela saída. Comandos do tipo `vi` também ajudam a navegar e pesquisar em todo o texto.

No entanto, em vez de canalizar o comando `cat` para um programa de paginação, é mais pragmático usar o programa de paginação diretamente:

```
$ sudo less /var/log/syslog
... (saída omitida para maior clareza)
```

Obtendo uma parte de um arquivo de texto

Se apenas o início ou o final de um arquivo precisar ser analisado, existem outros métodos disponíveis. O comando `head` é usado para ler as primeiras dez linhas de um arquivo por padrão, e o comando `tail` é usado para ler as dez linhas últimas. Experimente:

```
$ sudo head /var/log/syslog
Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd" swVersion="8.1910.0"
x-pid="811" x-info="https://www.rsyslog.com"] rsyslogd was HUPed
Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started (pid=882,
tid=928, prio=low)
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
```

```
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first
device!
Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882,
tid=929, prio=high)
$ sudo tail /var/log/syslog
Nov 13 10:24:45 hypatia kernel: [ 8001.679238] mce: CPU7: Core temperature/speed
normal
Nov 13 10:24:46 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Activating
via systemd: service name='org.freedesktop.Tracker1.Miner.Extract' unit='tracker-
extract.service' requested by ':1.73' (uid=1000 pid=2425
comm="/usr/lib/tracker/tracker-miner-fs ")
Nov 13 10:24:46 hypatia systemd[2004]: Starting Tracker metadata extractor...
Nov 13 10:24:47 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Successfully
activated service 'org.freedesktop.Tracker1.Miner.Extract'
Nov 13 10:24:47 hypatia systemd[2004]: Started Tracker metadata extractor.
Nov 13 10:24:54 hypatia kernel: [ 8010.462227] mce: CPU0: Core temperature above
threshold, cpu clock throttled (total events = 502907)
Nov 13 10:24:54 hypatia kernel: [ 8010.462228] mce: CPU4: Core temperature above
threshold, cpu clock throttled (total events = 502911)
Nov 13 10:24:54 hypatia kernel: [ 8010.469221] mce: CPU0: Core temperature/speed
normal
Nov 13 10:24:54 hypatia kernel: [ 8010.469222] mce: CPU4: Core temperature/speed
normal
Nov 13 10:25:03 hypatia systemd[2004]: tracker-extract.service: Succeeded.
```

Para ajudar a ilustrar o número de linhas exibidas, podemos canalizar a saída do comando `head` para o comando `nl`, que exibirá o número de linhas de texto transmitidas para o comando:

```
$ sudo head /var/log/syslog | nl
1   Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd"
swVersion="8.1910.0" x-pid="811" x-info="https://www.rsyslog.com"] rsyslogd was
HUPed
2   Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
3   Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
4   Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started
(pid=882, tid=928, prio=low)
5   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
6   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
7   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
8   Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
9   Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first
device!
10  Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882,
```

```
tid=929, prio=high)
```

E podemos fazer o mesmo canalizando a saída do comando `tail` para o comando `wc`, que por padrão conta o número de palavras em um documento, e usando a opção `-l` para imprimir na tela o número de linhas de texto lidas pelo comando:

```
$ sudo tail /var/log/syslog | wc -l
10
```

Caso um administrador precise revisar mais (ou menos) linhas do início ou do fim de um arquivo, a opção `-n` pode ser usada para limitar a saída dos comandos:

```
$ sudo tail -n 5 /var/log/syslog
Nov 13 10:37:24 hypatia systemd[2004]: tracker-extract.service: Succeeded.
Nov 13 10:37:42 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Activating
via systemd: service name='org.freedesktop.Tracker1.Miner.Extract' unit='tracker-
extract.service' requested by ':1.73' (uid=1000 pid=2425
comm="/usr/lib/tracker/tracker-miner-fs ")
Nov 13 10:37:42 hypatia systemd[2004]: Starting Tracker metadata extractor...
Nov 13 10:37:43 hypatia dbus-daemon[2023]: [session uid=1000 pid=2023] Successfully
activated service 'org.freedesktop.Tracker1.Miner.Extract'
Nov 13 10:37:43 hypatia systemd[2004]: Started Tracker metadata extractor.
$ sudo head -n 12 /var/log/syslog
Nov 12 08:04:30 hypatia rsyslogd: [origin software="rsyslogd" swVersion="8.1910.0"
x-pid="811" x-info="https://www.rsyslog.com"] rsyslogd was HUPed
Nov 12 08:04:30 hypatia systemd[1]: logrotate.service: Succeeded.
Nov 12 08:04:30 hypatia systemd[1]: Started Rotate log files.
Nov 12 08:04:30 hypatia vdr: [928] video directory scanner thread started (pid=882,
tid=928, prio=low)
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'A - ATSC'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'C - DVB-C'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'S - DVB-S'
Nov 12 08:04:30 hypatia vdr: [882] registered source parameters for 'T - DVB-T'
Nov 12 08:04:30 hypatia vdr[882]: vdr: no primary device found - using first
device!
Nov 12 08:04:30 hypatia vdr: [929] epg data reader thread started (pid=882,
tid=929, prio=high)
Nov 12 08:04:30 hypatia vdr: [882] no DVB device found
Nov 12 08:04:30 hypatia vdr: [882] initializing plugin: vnsiserver (1.8.0): VDR-
Network-Streaming-Interface (VNSI) Server
```

Noções básicas de sed, o editor de fluxo

Vamos dar uma olhada em outros arquivos, termos e utilitários que não têm `cat` em seus nomes. Podemos fazer isso passando a opção `-v` para `grep`, que instrui o comando a exibir apenas as linhas que não contêm `cat`:

```
$ zcat ftu.txt.gz | grep -v cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
```

Muito do que podemos fazer com `grep`, também podemos fazer com `sed` — o editor de fluxo para filtrar e transformar texto (como consta na página de manual do `sed`). Primeiro, vamos recuperar nosso arquivo `ftu.txt` descompactando o pacote `gzip` do arquivo:

```
$ gunzip ftu.txt.gz
$ ls ftu*
ftu.txt
```

Em seguida, podemos usar `sed` para listar apenas as linhas que contêm a string `cat`:

```
$ sed -n /cat/p < ftu.txt
bzcat
cat
xzcat
zcat
```

Usamos o sinal de menor que `<` para direcionar o conteúdo do arquivo `ftu.txt into` para o comando

`sed`. A palavra entre barras (isto é, `/cat/`) é o termo que estamos procurando. A opção `-n` instrui o `sed` a não produzir nenhuma saída (a não ser as saídas solicitadas pelo comando `p`). Tente executar este mesmo comando sem a opção `-n` para ver o que acontece. Depois, tente o seguinte:

```
$ sed /cat/d < ftu.txt
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
```

Sem a opção `-n`, `sed` imprime na tela tudo o que está no arquivo, exceto o que `d` instrui `sed` a remover da saída.

Um uso comum de `sed` é localizar e substituir texto em um arquivo. Suponha que você queira alterar todas as ocorrências de `cat` para `dog`. Para isso, use o `sed`, instruindo a opção `s` a trocar cada instância do primeiro termo, `cat`, pelo segundo, `dog`:

```
$ sed s/cat/dog/ < ftu.txt
bzdog
dog
ct
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
```

```
sort
split
tail
tr
uniq
wc
xzdog
zdog
```

Em vez de usar um operador de redirecionamento (`<`) para passar o arquivo `ftu.txt` para o comando `sed`, podemos fazer o comando `sed` operar diretamente no arquivo. Tentaremos isso a seguir, enquanto criamos simultaneamente um backup do arquivo original:

```
$ sed -i.backup s/cat/dog/ ftu.txt
$ ls ftu*
ftu.txt  ftu.txt.backup
```

A opção `-i` executa uma operação `sed` local no arquivo *original*. Se você não usar `.backup` após o parâmetro `-i`, vai reescrever o arquivo *original*. Qualquer texto inserido após o parâmetro `-i` será o nome com o qual o arquivo original será salvo antes das modificações que você pediu ao `sed` para realizar.

Garantindo a integridade dos dados

Como vimos, é fácil manipular arquivos no Linux. Há momentos em que queremos compartilhar um arquivo com outra pessoa e ter certeza de que o destinatário receberá uma cópia fiel do arquivo original. Um uso muito comum dessa técnica é quando os servidores das distribuições Linux hospedam imagens de CD ou DVD de seu software para download, juntamente com arquivos que contêm os valores calculados da soma de verificação (checksum) dessas imagens de disco. Eis, por exemplo, a listagem de um mirror de download do Debian:

```
                                                                                
[PARENTDIR] Parent Directory                                             -
[SUM]       MD5SUMS                                      2019-09-08 17:46 274
[CRT]       MD5SUMS.sign                                 2019-09-08 17:52 833
[SUM]       SHA1SUMS                                     2019-09-08 17:46 306
[CRT]       SHA1SUMS.sign                                2019-09-08 17:52 833
[SUM]       SHA256SUMS                                   2019-09-08 17:46 402
[CRT]       SHA256SUMS.sign                              2019-09-08 17:52 833
[SUM]       SHA512SUMS                                   2019-09-08 17:46 658
[CRT]       SHA512SUMS.sign                              2019-09-08 17:52 833
```

```
[ISO]       debian-10.1.0-amd64-netinst.iso      2019-09-08 04:37 335M
[ISO]       debian-10.1.0-amd64-xfce-CD-1.iso    2019-09-08 04:38 641M
[ISO]       debian-edu-10.1.0-amd64-netinst.iso  2019-09-08 04:38 405M
[ISO]       debian-mac-10.1.0-amd64-netinst.iso  2019-09-08 04:38 334M
```

Na listagem acima, os arquivos de imagem do instalador do Debian são acompanhados por arquivos de texto que contêm checksums dos arquivos dos vários algoritmos (MD5, SHA1, SHA256 e SHA512).

NOTE

Uma soma de verificação (checksum) é um valor derivado de um cálculo matemático, baseado em uma função hash criptográfica, em relação a um arquivo. Existem diferentes tipos de funções hash criptográficas com diferentes intensidades. Para o exame, você deverá estar familiarizado com o uso de `md5sum`, `sha256sum` e `sha512sum`.

Depois de baixar um arquivo (por exemplo, a imagem `debian-10.1.0-amd64-netinst.iso`), você deve comparar a soma de verificação do arquivo baixado com o valor de soma de verificação que lhe foi fornecido.

Eis um exemplo ilustrativo. Vamos calcular o valor SHA256 do arquivo `ftu.txt` usando o comando `sha256sum`:

```
$ sha256sum ftu.txt
345452304fc26999a715652543c352e5fc7ee0c1b9deac6f57542ec91daf261c  ftu.txt
```

A longa sequência de caracteres que precede o nome do arquivo é o valor do checksum SHA256 desse arquivo de texto. Vamos criar um arquivo contendo esse valor, que servirá para verificar a integridade do arquivo de texto original. Usamos para isso o mesmo comando `sha256sum` e redirecionamos a saída para um arquivo:

```
$ sha256sum ftu.txt > sha256.txt
```

Agora, para verificar o arquivo `ftu.txt`, basta usar o mesmo comando e fornecer o nome do arquivo que contém o valor do checksum junto com a opção `-c`:

```
$ sha256sum -c sha256.txt
ftu.txt: OK
```

O valor contido no arquivo corresponde à soma de verificação SHA256 calculada para nosso arquivo `ftu.txt`, exatamente como esperávamos. No entanto, se o arquivo original for modificado (por exemplo, alguns bytes perdidos durante o download, ou uma adulteração deliberada), a verificação do valor falhará. Nesse caso, sabemos que o arquivo está danificado ou corrompido e não podemos confiar na integridade de seu conteúdo. Para demonstrar esse ponto, vamos adicionar algum texto no final do arquivo:

```
$ echo "new entry" >> ftu.txt
```

A seguir, tentaremos verificar a integridade do arquivo:

```
$ sha256sum -c sha256.txt
ftu.txt: FAILED
sha256sum: WARNING: 1 computed checksum did NOT match
```

Vemos assim que a soma de verificação não corresponde ao esperado. Portanto, não podemos confiar na integridade deste arquivo. Poderíamos tentar baixar uma nova cópia, relatar a falha da soma de verificação ao remetente do arquivo ou contatar a equipe de segurança do data center, dependendo da importância do arquivo.

Um olhar mais aprofundado nos arquivos

O comando octal dump (`od`) é freqüentemente usado para depurar aplicativos e diversos tipos de arquivos. Por si só, o comando `od` somente lista o conteúdo de um arquivo em formato octal. Podemos usar nosso arquivo `ftu.txt` para praticar com esse comando:

```
$ od ftu.txt
0000000 075142 060543 005164 060543 005164 072543 005164 062550
0000020 062141 066012 071545 005163 062155 071465 066565 067012
0000040 005154 062157 070012 071541 062564 071412 062145 071412
0000060 060550 032462 071466 066565 071412 060550 030465 071462
0000100 066565 071412 071157 005164 070163 064554 005164 060564
0000120 066151 072012 005162 067165 070551 073412 005143 075170
0000140 060543 005164 061572 072141 000012
0000151
```

A primeira coluna é o *deslocamento de bytes* para cada linha da saída. Como por padrão `od` imprime informações no formato octal, cada linha começa com um deslocamento de bytes de oito bits, seguido por oito colunas, cada uma contendo o valor octal dos dados dentro dessa coluna.

TIP Lembre que um *byte* tem 8 bits.

Se precisar visualizar o conteúdo de um arquivo em formato hexadecimal, use a opção `-x`:

```
$ od -x ftu.txt
0000000 7a62 6163 0a74 6163 0a74 7563 0a74 6568
0000020 6461 6c0a 7365 0a73 646d 7335 6d75 6e0a
0000040 0a6c 646f 700a 7361 6574 730a 6465 730a
0000060 6168 3532 7336 6d75 730a 6168 3135 7332
0000100 6d75 730a 726f 0a74 7073 696c 0a74 6174
0000120 6c69 740a 0a72 6e75 7169 770a 0a63 7a78
0000140 6163 0a74 637a 7461 000a
0000151
```

Nesse caso, cada uma das oito colunas após o deslocamento de bytes é representada por seus equivalentes hexadecimais.

Um uso útil do comando `od` é depurar scripts. Por exemplo, o comando `od` pode nos mostrar caracteres que não são normalmente visíveis e que existem em um arquivo, como indicadores de *nova linha*. Usamos para isso a opção `-c`, de modo que, em vez de exibir a notação numérica para cada byte, as colunas serão mostradas como seus equivalentes em caracteres:

```
$ od -c ftu.txt
0000000   b   z   c   a   t  \n   c   a   t  \n   c   u   t  \n   h   e
0000020   a   d  \n   l   e   s   s  \n   m   d   5   s   u   m  \n   n
0000040   l  \n   o   d  \n   p   a   s   t   e  \n   s   e   d  \n   s
0000060   h   a   2   5   6   s   u   m  \n   s   h   a   5   1   2   s
0000100   u   m  \n   s   o   r   t  \n   s   p   l   i   t  \n   t   a
0000120   i   l  \n   t   r  \n   u   n   i   q  \n   w   c  \n   x   z
0000140   c   a   t  \n   z   c   a   t  \n
0000151
```

Todas as ocorrências de nova linha no arquivo são representadas pelos caracteres ocultos `\n`. Se você deseja apenas visualizar todos os caracteres em um arquivo e não precisa ver as informações de deslocamento de bytes, pode remover essa coluna da saída da seguinte forma:

```
$ od -An -c ftu.txt
   b   z   c   a   t  \n   c   a   t  \n   c   u   t  \n   h   e
   a   d  \n   l   e   s   s  \n   m   d   5   s   u   m  \n   n
   l  \n   o   d  \n   p   a   s   t   e  \n   s   e   d  \n   s
   h   a   2   5   6   s   u   m  \n   s   h   a   5   1   2   s
```

```
u  m  \n  s  o  r  t  \n  s  p  l  i  t  \n  t  a
i  l  \n  t  r  \n  u  n  i  q  \n  w  c  \n  x  z
c  a  t  \n  z  c  a  t  \n
```

Exercícios Guiados

1. Alguém acaba de doar um laptop para sua escola e você deseja instalar Linux nele. Ele veio sem manual e você é obrigado a inicializá-lo a partir de um pen drive USB sem nenhuma interface gráfica. Aparece um terminal shell e você sabe que, para cada processador presente, haverá uma linha dedicada no arquivo `/proc/cpuinfo`:

```
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(linhas puladas)

processor   : 1
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(mais linhas puladas)
```

   - Usando os comandos `grep` e `wc`, exiba o número de processadores presentes.

   - Faça o mesmo com `sed` em vez de `grep`.

2. Explore seu arquivo local `/etc/passwd` com os comandos `grep`, `sed`, `head` e `tail` de acordo com as tarefas descritas abaixo:

   - Quais usuários têm acesso a um shell Bash?

   - Muitos dos usuários de seu sistema existem para lidar com programas específicos ou para fins administrativos. Eles não têm acesso a um shell. Quantos deles estão presentes no sistema?

   - Quantos usuários e grupos existem em seu sistema (lembre-se: use apenas o arquivo `/etc/passwd`)?

- Liste apenas a primeira, a última e a décima linha do arquivo `/etc/passwd`.

3. Considere este arquivo de exemplo `/etc/passwd`. `Copie as linhas abaixo para um arquivo local chamado `mypasswd` para este exercício.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
nvidia-persistenced:x:121:128:NVIDIA Persistence
Daemon,,,:/nonexistent:/sbin/nologin
libvirt-qemu:x:64055:130:Libvirt Qemu,,,:/var/lib/libvirt:/usr/sbin/nologin
libvirt-dnsmasq:x:122:133:Libvirt
Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/usr/sbin/nologin
carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
dave:x:1001:1000:Dave Edwards,Finance,,,Main Office:/home/dave:/bin/ksh
emma:x:1002:1000:Emma Jones,Finance,,,Main Office:/home/emma:/bin/bash
frank:x:1003:1000:Frank Cassidy,Finance,,,Main Office:/home/frank:/bin/bash
grace:x:1004:1000:Grace Kearns,Engineering,,,Main Office:/home/grace:/bin/ksh
henry:x:1005:1000:Henry Adams,Sales,,,Main Office:/home/henry:/bin/bash
john:x:1006:1000:John Chapel,Sales,,,Main Office:/home/john:/bin/bash
```

- Liste todos os usuários no grupo `1000` (use `sed` para selecionar apenas o campo apropriado) do arquivo `mypasswd`.

- Liste apenas o nome completo de todos os usuários deste grupo (use `sed` e `cut`).

Exercícios Exploratórios

1. Usando novamente o arquivo `mypasswd` dos exercícios anteriores, imagine um comando Bash que selecione um indivíduo do Main Office para ganhar uma rifa. Use o comando `sed` para imprimir apenas as linhas do Main Office e, em seguida, uma sequência de comando `cut` para recuperar o primeiro nome de cada usuário a partir dessas linhas. Depois, classifique esses nomes aleatoriamente e imprima apenas o nome principal da lista.

2. Quantas pessoas trabalham em Finance, Engineering e Sales? Pense em explorar o comando `uniq`.

3. Agora, vamos preparar um arquivo CSV (valores separados por vírgula) para poder importar facilmente, do arquivo `mypasswd` do exemplo anterior, o arquivo `names.csv` para o LibreOffice. O conteúdo do arquivo terá o seguinte formato:

   ```
   First Name,Last Name,Position
   Carol,Smith,Finance
   ...
   John,Chapel,Sales
   ```

   Dica: use os comandos `sed`, `cut` e `paste` para obter os resultados desejados. Note que a vírgula (`,`) será o delimitador desse arquivo.

4. Suponha que a planilha `names.csv` criada no exercício anterior seja um arquivo importante e queremos ter certeza de que ninguém vai adulterá-lo desde o momento do envio até a recepção pelo destinatário. Como podemos garantir a integridade desse arquivo usando `md5sum`?

5. Você prometeu a si mesmo que leria 100 linhas por dia de um livro clássico e decidiu começar com *Mariner and Mystic* de Herman Melville. Desenvolva um comando usando `split` para dividir este livro em seções de 100 linhas cada. Para obter o livro em formato de texto simples, pesquise em https://www.gutenberg.org.

6. Usando `ls -l` no diretório `/etc`, que tipo de listagem obtemos? Usando o comando `cut` na saída do comando `ls` fornecido, como exibir apenas os nomes dos arquivos? E quanto ao nome dos arquivos e seu proprietário? Junto com os comandos `ls -l` e `cut`, utilize o comando `tr` para

espremer diversas ocorrências de um espaço em um único espaço para auxiliar na formatação da saída com um comando `cut`.

7. Este exercício pressupõe que você está em uma máquina real (não virtual). Também é preciso ter um pendrive à mão. Releia as páginas de manual do comando `tail` e descubra como seguir um arquivo conforme adicionamos texto a ele. Enquanto monitora a saída de um comando `tail` no arquivo `/var/log/syslog`, insira um pendrive. Escreva o comando completo que usaria para obter o Produto (Product), o Fabricante (Manufacturer) e a quantidade total de memória (Blocks) do seu pendrive.

Resumo

Lidar com fluxos de texto é de grande importância ao administrar qualquer sistema Linux. Os fluxos de texto podem ser processados usando scripts para automatizar tarefas diárias ou localizar informações de depuração relevantes em arquivos de log. Eis um breve resumo dos comandos abordados nesta lição:

`cat`

Usado para combinar ou ler arquivos de texto simples.

`bzcat`

Permite processar ou ler arquivos comprimidos usando o método `bzip2`.

`xzcat`

Permite processar ou ler arquivos comprimidos usando o método `xz`.

`zcat`

Permite processar ou ler arquivos comprimidos usando o método `gzip`.

`less`

Este comando pagina o conteúdo de um arquivo e possibilita a funcionalidade de navegação e pesquisa.

`head`

Este comando exibe as primeiras 10 linhas de um arquivo por padrão. A opção `-n` permite exibir mais ou menos linhas.

`tail`

Este comando exibe as últimas 10 linhas de um arquivo por padrão. A opção `-n` permite exibir mais ou menos linhas. A opção `-f` é usada para seguir a saída de um arquivo de texto conforme novos dados são escritos nele.

`wc`

Abreviação de "word count" (contagem de palavras), mas dependendo dos parâmetros usados ele conta caracteres, palavras e linhas.

`sort`

Usado para organizar a saída de uma listagem em ordem alfabética, alfabética inversa ou aleatória.

`uniq`

Usado para listar (e contar) strings correspondentes.

`od`

O comando "octal dump" é usado para exibir um arquivo binário em notação octal, decimal ou hexadecimal.

`nl`

O comando "number line" exibe o número de linhas em um arquivo, além de recriar um arquivo com cada linha prefixada por seu número de linha.

`sed`

O editor de fluxo pode ser usado para encontrar ocorrências correspondentes de strings usando Expressões Regulares, bem como editar arquivos usando padrões predefinidos.

`tr`

O comando translate substitui caracteres e também remove e compacta caracteres repetidos.

`cut`

Este comando imprime colunas de arquivos de texto como campos baseados no delimitador de caracteres de um arquivo.

`paste`

Une arquivos em colunas com base no uso de separadores de campo.

`split`

Este comando divide arquivos maiores em menores conforme os critérios definidos nas opções do comando.

`md5sum`

Usado para calcular o valor de hash MD5 de um arquivo. Também usado para verificar um arquivo em relação a um valor de hash existente para garantir a integridade do arquivo.

`sha256sum`

Usado para calcular o valor de hash SHA256 de um arquivo. Também usado para verificar um arquivo em relação a um valor de hash existente para garantir a integridade do arquivo.

`sha512sum`

Usado para calcular o valor de hash SHA512 de um arquivo. Também usado para verificar um

arquivo em relação a um valor de hash existente para garantir a integridade do arquivo.

Respostas aos Exercícios Guiados

1. Alguém acaba de doar um laptop para sua escola e você deseja instalar Linux nele. Ele veio sem manual e você é obrigado a inicializá-lo a partir de um pen drive USB sem nenhuma interface gráfica. Aparece um terminal shell e você sabe que, para cada processador presente, haverá uma linha dedicada no arquivo `/proc/cpuinfo`:

```
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(linhas puladas)

processor   : 1
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(more lines skipped)
```

   - Usando os comandos `grep` e `wc`, exiba o número de processadores presentes.

     Eis duas opções:

```
$ cat /proc/cpuinfo | grep processor | wc -l
$ grep processor /proc/cpuinfo | wc -l
```

     Agora que você sabe que existem várias maneiras de fazer a mesma coisa, quando deve usar uma ou outra? Isso depende de vários fatores, sendo os dois mais importantes o desempenho e a legibilidade. Na maioria das vezes, usamos comandos de shell dentro de scripts de shell para automatizar tarefas; quanto maiores e mais complexos seus scripts se tornam, mais precisamos nos preocupar em mantê-los rápidos.

   - Faça o mesmo com `sed` em vez de `grep`.

     Agora, em vez de `grep` vamos tentar isso com `sed`:

```
$ sed -n /processor/p /proc/cpuinfo | wc -l
```

Aqui, usamos `sed` com o parâmetro `-n` para que `sed` não imprima nada, exceto o que corresponde à expressão `processor`, conforme instruído pelo comando `p` . Como fizemos nas soluções de `grep`, `wc -l` conta o número de linhas e, portanto, o número de processadores presentes.

Estude o exemplo a seguir:

```
$ sed -n /processor/p /proc/cpuinfo | sed -n '$='
```

Esta sequência de comando fornece resultados idênticos aos do exemplo anterior, no qual a saída de `sed` foi canalizada para um comando `wc`. A diferença, aqui, é que em vez de usar `wc -l` para contar o número de linhas, `sed` é novamente invocado para fornecer uma funcionalidade equivalente. Mais uma vez, estamos suprimindo a saída de `sed` com a opção `-n`, exceto para a expressão que estamos chamando explicitamente, que é `'$='`. Essa expressão diz ao `sed` para encontrar uma correspondência para a última linha (`$`) e então imprimir o número dessa linha (`=`).

2. Explore seu arquivo local `/etc/passwd` com os comandos `grep`, `sed`, `head` e `tail` de acordo com as tarefas descritas abaixo:
   - Quais usuários têm acesso a um shell Bash?

     ```
     $ grep ":/bin/bash$" /etc/passwd
     ```

     Vamos refinar esta resposta exibindo apenas o nome do usuário que utiliza o shell Bash.

     ```
     $ grep ":/bin/bash$" /etc/passwd | cut -d: -f1
     ```

     O nome do usuário é o primeiro campo (parâmetro `-f1` do comando `cut`) e o arquivo `/etc/passwd` usa `:` como separadores (`-d:` parâmetro do comando `cut`). Nós apenas canalizamos a saída do comando `grep` para o comando `cut` apropriado.

   - Muitos dos usuários de seu sistema existem para lidar com programas específicos ou para fins administrativos. Eles não têm acesso a um shell. Quantos deles estão presentes no sistema?

     A maneira mais fácil de descobrir isso é exibir as linhas correspondentes às contas que não usam o shell Bash:

```
$ grep -v ":/bin/bash$" /etc/passwd | wc -l
```

- Quantos usuários e grupos existem em seu sistema (lembre-se: use apenas o arquivo `/etc/passwd`)?

  O primeiro campo de qualquer linha do arquivo `/etc/passwd` é o nome do usuário, o segundo é tipicamente um `x` indicando que a senha do usuário não está armazenada aqui (ela é criptografada no arquivo `/etc/shadow`). O terceiro é o id do usuário (UID) e o quarto é o id do grupo (GID). Portanto, isso deve nos fornecer o número de usuários:

  ```
  $ cut -d: -f3 /etc/passwd | wc -l
  ```

  Bem, isso funciona na maioria dos casos. No entanto, há situações em que definimos diferentes superusuários ou outros tipos especiais de usuários que compartilham o mesmo UID (id de usuário). Assim, para ter certeza, vamos canalizar o resultado do comando `cut` para o comando `sort` e então contar o número de linhas.

  ```
  $ cut -d: -f3 /etc/passwd | sort -u | wc -l
  ```

  Agora, para o número de grupos:

  ```
  $ cut -d: -f4 /etc/passwd | sort -u | wc -l
  ```

- Liste apenas a primeira, a última e a décima linha do arquivo `/etc/passwd`.

  Podemos fazer assim:

  ```
  $ sed -n -e '1'p -e '10'p -e '$'p /etc/passwd
  ```

  Lembre-se de que o parâmetro `-n` diz ao `sed` para não imprimir nada além do que é especificado pelo comando `p`. O cifrão (`$`) usado aqui é uma expressão regular que representa a última linha do arquivo.

3. Considere este exemplo do arquivo `/etc/passwd`. Copie as linhas abaixo para um arquivo local chamado `mypasswd` para este exercício.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
nvidia-persistenced:x:121:128:NVIDIA Persistence
Daemon,,,:/nonexistent:/sbin/nologin
libvirt-qemu:x:64055:130:Libvirt Qemu,,,:/var/lib/libvirt:/usr/sbin/nologin
libvirt-dnsmasq:x:122:133:Libvirt
Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/usr/sbin/nologin
carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
dave:x:1001:1000:Dave Edwards,Finance,,,Main Office:/home/dave:/bin/ksh
emma:x:1002:1000:Emma Jones,Finance,,,Main Office:/home/emma:/bin/bash
frank:x:1003:1000:Frank Cassidy,Finance,,,Main Office:/home/frank:/bin/bash
grace:x:1004:1000:Grace Kearns,Engineering,,,Main Office:/home/grace:/bin/ksh
henry:x:1005:1000:Henry Adams,Sales,,,Main Office:/home/henry:/bin/bash
john:x:1006:1000:John Chapel,Sales,,,Main Office:/home/john:/bin/bash
```

- Liste todos os usuários no grupo `1000` (use `sed` para selecionar apenas o campo apropriado) do arquivo `mypasswd`.

  O GID é o quarto campo do arquivo `/etc/passwd`. Você pode ter vontade de tentar o seguinte:

  ```
  $ sed -n /1000/p mypasswd
  ```

  Nesse caso, você obterá também esta linha:

  ```
  carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
  ```

  Sabemos que não está correto, pois Carol Smith é membro do GID 2000 e a correspondência foi encontrada por causa do UID. No entanto, você deve ter notado que, após o GID, o campo seguinte começa com um caractere maiúsculo. Podemos usar uma expressão regular para resolver esse problema.

  ```
  $ sed -n /:1000:[A-Z]/p mypasswd
  ```

  A expressão `[A-Z]` busca por quaisquer caracteres em maiúsculas. Vamos falar mais sobre isso na lição correspondente.

◦ Liste apenas o nome completo de todos os usuários deste grupo (use `sed` e `cut`).

Use a mesma técnica empregada para resolver a primeira parte deste exercício e canalize para um comando `cut`.

```
$ sed -n /:1000:[A-Z]/p mypasswd | cut -d: -f5
Dave Edwards,Finance,,,Main Office
Emma Jones,Finance,,,Main Office
Frank Cassidy,Finance,,,Main Office
Grace Kearns,Engineering,,,Main Office
Henry Adams,Sales,,,Main Office
John Chapel,Sales,,,Main Office
```

Ainda não chegamos lá! Note como os campos dentro dos resultados podem ser separados por `,`. Assim, vamos canalizar a saída para outro comando `cut`, usando `,` como delimitador.

```
$ sed -n /:1000:[A-Z]/p mypasswd | cut -d: -f5 | cut -d, -f1
Dave Edwards
Emma Jones
Frank Cassidy
Grace Kearns
Henry Adams
John Chapel
```

Respostas aos Exercícios Exploratórios

1. Usando novamente o arquivo `mypasswd` dos exercícios anteriores, imagine um comando Bash que selecione um indivíduo do Main Office para ganhar uma rifa. Use o comando `sed` para imprimir apenas as linhas do Main Office e, em seguida, uma sequência de comando `cut` para recuperar o primeiro nome de cada usuário a partir dessas linhas. Depois, classifique esses nomes aleatoriamente e imprima apenas o nome principal da lista.

   Primeiro, explore como o parâmetro `-R` manipula a saída do comando `sort`. Repita esse comando algumas vezes em sua máquina (note que será preciso colocare 'Main Office' entre aspas simples para que o `sed` o trate como uma única string):

   ```
   $ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1 | sort -R
   ```

   Eis uma solução para o problema:

   ```
   $ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1 | sort -R | head
   -1
   ```

2. Quantas pessoas trabalham em Finance, Engineering e Sales? Pense em explorar o comando `uniq`.

   Continue incrementando o que você aprendeu nos exercícios anteriores. Tente o seguinte:

   ```
   $ sed -n /'Main Office'/p mypasswd
   $ sed -n /'Main Office'/p mypasswd | cut -d, -f2
   ```

   Note que não nos preocupamos com o delimitador `:`. Queremos somente o segundo campo quando dividimos as linhas pelos caracteres `,`.

   ```
   $ sed -n /'Main Office'/p mypasswd | cut -d, -f2 | uniq -c
         4 Finance
         1 Engineering
         2 Sales
   ```

   O comando `uniq` mostra apenas as linhas únicas (não as linhas repetidas) e o parâmetro `-c` diz ao `uniq` para contar as ocorrências de linhas iguais. Mas fique esperto: `uniq` considera apenas as linhas adjacentes. Quando esse não for o caso, será preciso usar o comando `sort`.

3. Agora, vamos preparar um arquivo CSV (valores separados por vírgula) para poder importar facilmente, do arquivo `mypasswd` do exemplo anterior, o arquivo `names.csv` para o LibreOffice. O conteúdo do arquivo terá o seguinte formato:

```
First Name,Last Name,Position
Carol,Smith,Finance
...
John,Chapel,Sales
```

Dica: use os comandos `sed`, `cut` e `paste` para obter os resultados desejados. Note que a vírgula (`,`) será o delimitador desse arquivo.

Comece com os comandos `sed` e `cut`, incrementando o que aprendemos nos exercícios anteriores:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d" " -f1 > firstname
```

Agora temos o arquivo `firstname` com o primeiro nome dos funcionários.

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d" " -f2 | cut -d, -f1
> lastname
```

Agora temos o arquivo `lastname` com o sobrenome de cada funcionário.

A seguir, vamos determinar o departamento em que cada funcionário trabalha:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f2 > department
```

Antes de trabalharmos na resposta final, experimente os seguintes comandos para ver que tipo de saída eles geram:

```
$ cat firstname lastname department
$ paste firstname lastname department
```

E a solução é:

```
$ paste firstname lastname department | tr '\t' ,
```

```
$ paste firstname lastname department | tr '\t' , > names.csv
```

Aqui, usamos o comando `tr` para *traduzir* `\t`, o separador de tabulação, por um `,`. `tr` é bastante útil quando precisamos trocar um caractere por outro. Não deixe de ler as páginas de manual de `tr` e `paste`. Por exemplo, podemos usar a opção `-d` para o delimitador, para tornar o comando anterior menos complexo:

```
$ paste -d, firstname lastname department
```

Usamos o comando `paste` aqui para você se acostumar melhor com ele. No entanto, poderíamos ter executado facilmente todas as tarefas em uma única cadeia de comando:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1,2 | tr ' ' , >
names.csv
```

4. Suponha que a planilha `names.csv` criada no exercício anterior seja um arquivo importante e queremos ter certeza de que ninguém vai adulterá-lo desde o momento do envio até a recepção pelo destinatário. Como podemos garantir a integridade desse arquivo usando `md5sum`?

   Se você consultar as páginas de manual de `md5sum`, `sha256sum` e `sha512sum`, verá que todas começam com o seguinte texto:

   "compute and check XXX message digest"

   Onde "XXX" é o algoritmo que será usado para criar esse *message digest* (resumo de mensagens).

   Usaremos `md5sum` como exemplo e, mais tarde, você poderá tentar com os outros comandos.

```
$ md5sum names.csv
61f0251fcab61d9575b1d0cbf0195e25  names.csv
```

   Agora, por exemplo, você pode disponibilizar o arquivo através de um serviço de ftp seguro e enviar o *resumo de mensagens* gerado usando outro meio de comunicação seguro. Se o arquivo tiver sido ligeiramente alterado, o *resumo de mensagens* será totalmente diferente. Para comprovar, edite `names.csv` e troque Jones por James, como demonstramos aqui:

```
$ sed -i.backup s/Jones/James/ names.csv
$ md5sum names.csv
```

```
f44a0d68cb480466099021bf6d6d2e65  names.csv
```

Sempre que você disponibilizar arquivos para download, é aconselhável distribuir também um *resumo de mensagens* correspondente para que as pessoas que baixarem aquele arquivo possam produzir um novo *resumo de mensagens* e comparar com o original. Se você visitar o site https://kernel.org, encontrará a página https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/sha256sums.asc, onde pode obter o sha256sum de todos os arquivos disponíveis para download.

5. Você prometeu a si mesmo que leria 100 linhas por dia de um livro clássico e decidiu começar com *Mariner and Mystic* de Herman Melville. Desenvolva um comando usando `split` para dividir este livro em seções de 100 linhas cada. Para obter o livro em formato de texto simples, pesquise em https://www.gutenberg.org.

   Primeiro, baixamos o livro completo do site do Project Gutenberg, onde você pode obter este e outros livros disponíveis em domínio público.

   ```
   $ wget https://www.gutenberg.org/files/50461/50461-0.txt
   ```

   Pode ser preciso instalar `wget` se ainda não estiver presente no sistema. Outra alternativa é usar `curl`. Use `less` para verificar o livro:

   ```
   $ less 50461-0.txt
   ```

   Agora, vamos dividir o livro em trechos de 100 linhas cada:

   ```
   $ split -l 100 -d 50461-0.txt melville
   ```

   `50461-0.txt` é o arquivo que dividiremos. `melville` será o prefixo dos arquivos divididos. `-l 100` especifica o número de linhas e a opção `-d` diz ao `split` para numerar os arquivos (usando o sufixo fornecido). Podemos usar `nl` em qualquer um dos arquivos divididos (provavelmente não no último) e confirmar que todos eles têm 100 linhas.

6. Usando `ls -l` no diretório `/etc`, que tipo de listagem obtemos? Usando o comando `cut` na saída do comando `ls` fornecido, como exibir apenas os nomes dos arquivos? E quanto ao nome dos arquivos e seu proprietário? Junto com os comandos `ls -l` e `cut`, utilize o comando `tr` para *espremer* diversas ocorrências de um espaço em um único espaço para auxiliar na formatação da saída com um comando `cut`.

O comando `ls` sozinho fornece apenas os nomes dos arquivos. Podemos, no entanto, preparar a saída do `ls -l` (a lista longa) para extrair informações mais específicas.

```
$ ls -l /etc | tr -s ' ' ,
drwxr-xr-x,3,root,root,4096,out,24,16:58,acpi
-rw-r--r--,1,root,root,3028,dez,17,2018,adduser.conf
-rw-r--r--,1,root,root,10,out,2,17:38,adjtime
drwxr-xr-x,2,root,root,12288,out,31,09:40,alternatives
-rw-r--r--,1,root,root,401,mai,29,2017,anacrontab
-rw-r--r--,1,root,root,433,out,1,2017,apg.conf
drwxr-xr-x,6,root,root,4096,dez,17,2018,apm
drwxr-xr-x,3,root,root,4096,out,24,16:58,apparmor
drwxr-xr-x,9,root,root,4096,nov,6,20:20,apparmor.d
```

O parâmetro `-s` instrui `tr` a reduzir os espaços repetidos a uma única instância de um espaço. O comando `tr` funciona para qualquer tipo de caractere repetitivo que você especificar. Em seguida, substituímos os espaços por uma vírgula `,`. Na verdade, não precisamos substituir os espaços em nosso exemplo, então apenas omitiremos `,`.

```
$ ls -l /etc | tr -s ' '
drwxr-xr-x 3 root root 4096 out 24 16:58 acpi
-rw-r--r-- 1 root root 3028 dez 17 2018 adduser.conf
-rw-r--r-- 1 root root 10 out 2 17:38 adjtime
drwxr-xr-x 2 root root 12288 out 31 09:40 alternatives
-rw-r--r-- 1 root root 401 mai 29 2017 anacrontab
-rw-r--r-- 1 root root 433 out 1 2017 apg.conf
drwxr-xr-x 6 root root 4096 dez 17 2018 apm
drwxr-xr-x 3 root root 4096 out 24 16:58 apparmor
```

Se quisermos apenas os nomes dos arquivos, só precisamos exibir o nono campo:

```
$ ls -l /etc | tr -s ' ' | cut -d" " -f9
```

Para o nome de arquivo e seu proprietário, precisamos do nono e do terceiro campos:

```
$ ls -l /etc | tr -s ' ' | cut -d" " -f9,3
```

E se só precisarmos dos nomes das pastas e seu proprietário?

```
$ ls -l /etc | grep ^d | tr -s ' ' | cut -d" " -f9,3
```

7. Este exercício pressupõe que você está em uma máquina real (não virtual). Também é preciso ter um pendrive à mão. Releia as páginas de manual do comando `tail` e descubra como seguir um arquivo conforme adicionamos texto a ele. Enquanto monitora a saída de um comando `tail` no arquivo `/var/log/syslog`, insira um pendrive. Escreva o comando completo que usaria para obter o Produto (Product), o Fabricante (Manufacturer) e a quantidade total de memória (Blocks) do seu pendrive.

```
$ tail -f /var/log/syslog | grep -i 'product\:\|blocks\|manufacturer'
Nov  8 06:01:35 brod-avell kernel: [124954.369361] usb 1-4.3: Product: Cruzer
Blade
Nov  8 06:01:35 brod-avell kernel: [124954.369364] usb 1-4.3: Manufacturer:
SanDisk
Nov  8 06:01:37 brod-avell kernel: [124955.419267] sd 2:0:0:0: [sdc] 61056064
512-byte logical blocks: (31.3 GB/29.1 GiB)
```

   Claro que isto é um exemplo e os resultados podem variar dependendo do fabricante do seu pendrive. Observe que agora usamos o parâmetro `-i` com o comando `grep`, pois não sabemos se as strings que procuramos estão em maiúsculas ou minúsculas. Também usamos `|` como um OR (ou) lógico, então procuramos por linhas contendo `product` OR `blocks` OR `manufacturer`.