

# Introdução ao Shell para Bioinformática

Izinara Rosse da Cruz

`izinara.cruz@ufop.edu.br`

Lauro Ângelo Gonçalves de Moraes

`lauromoraes@ufop.edu.br`

2022

# Sumário

<b>1</b>	<b>Introdução ao Unix</b>	<b>5</b>
1.1	Como o Shell se compara a uma interface de desktop? .	5
<b>2</b>	<b>Manipulando arquivos e diretórios</b>	<b>7</b>
2.1	Onde estou? . . . . .	8
2.2	Como posso identificar arquivos e diretórios? . . . . .	9
2.3	Como dar nomes aos meus arquivos? . . . . .	10
2.4	De que outra forma posso identificar meus arquivos e diretórios? . . . . .	12
2.5	Como posso mover para outro diretório? . . . . .	14
2.6	Como posso subir um diretório? . . . . .	14
2.7	Como posso copiar arquivos? . . . . .	16
2.8	Como posso mover um arquivo? . . . . .	17
2.9	Como posso renomear arquivos? . . . . .	18
2.10	Como posso deletar arquivos? . . . . .	19
2.11	Como posso criar e excluir diretórios? . . . . .	20
2.12	Como criar novos arquivos? . . . . .	21

2.13 Juntando tudo . . . . .	22
<b>3 Manipulando dados</b>	<b>24</b>
3.1 Como posso ver o conteúdo de um arquivo? . . . . .	24
3.2 Como posso ver o conteúdo de um arquivo pedaço por pedaço? . . . . .	25
3.3 Como posso ver o início de um arquivo? . . . . .	26
3.4 Como posso controlar o que os comandos fazem? . . . . .	27
3.5 Como posso listar tudo abaixo de um diretório? . . . . .	28
3.6 Como posso obter ajuda para um comando? . . . . .	29
3.7 Como posso seleccionar colunas de um arquivo? . . . . .	30
3.8 O que <b>cut</b> não pode fazer? . . . . .	31
3.9 Como posso repetir comandos? . . . . .	32
3.10 Como posso seleccionar linhas contendo valores específi- cos? . . . . .	33
3.11 Por que nem sempre é seguro tratar os dados como texto?	35
<b>4 Combinando ferramentas</b>	<b>37</b>
4.1 Como armazenar a saída de um comando em um arquivo?	37

4.2	Como posso usar a saída de um comando como entrada de outro? . . . . .	38
4.3	Qual é a melhor maneira de combinar comandos? . . . .	39
4.4	Como posso combinar vários comandos? . . . . .	40
4.5	Como posso contar os registros em um arquivo? . . . . .	41
4.6	Como posso especificar muitos arquivos de uma só vez?	42
4.7	Que outros curingas posso usar? . . . . .	43
4.8	Como posso ordenar as linhas de texto? . . . . .	44
4.9	Como posso remover linhas duplicadas? . . . . .	45
4.10	Como posso salvar a saída de um <i>pipe</i> ? . . . . .	47
4.11	Como posso interromper um programa em execução? . .	48
4.12	Juntando tudo . . . . .	48

# 1 Introdução ao Unix

A interface de linha de comando sobrevive e prospera por mais de 50 anos. Ela permite que as pessoas façam coisas complexas com apenas alguns toques no teclado. Ela permite combinar programas de novas maneiras, automatizar tarefas repetitivas e executar programas em clusters e nuvens que podem estar do outro lado do mundo. Este material apresentará uma introdução a seus elementos-chave e mostrará como usá-los de maneira eficaz.

## 1.1 Como o Shell se compara a uma interface de desktop?

Um sistema operacional como Windows, Linux ou Mac OS é um tipo especial de programa. Ele controla o processador do computador, o disco rígido e a conexão de rede, mas sua tarefa mais importante é executar outros programas.

Como os seres humanos não são digitais, eles precisam de uma interface para interagir com o sistema operacional. O mais comum hoje em dia é um explorador gráfico de arquivos, que traduz cliques e cliques duplos em comandos para abrir arquivos e executar programas. Antes dos computadores terem telas gráficas, porém, as pessoas digitavam instruções em um programa chamado Shell de linha de comando. Cada vez que um comando é inserido, o Shell executa alguns outros programas, imprime sua saída em formato legível e exibe um prompt para sinalizar que está pronto para aceitar o próximo comando. (Seu

nome vem da noção de que é a "camada externa" do computador.)

Digitar comandos em vez de clicar e arrastar pode parecer desajeitado no início, mas como você verá, depois de começar a soletrar o que deseja que o computador faça, você pode combinar comandos antigos para criar novos e automatizar operações repetitivas com apenas algumas teclas .

### Pergunta

Qual é a relação entre o explorador gráfico de arquivos que a maioria das pessoas usa e o Shell da linha de comando?

- A O explorador de arquivos permite que você visualize e edite arquivos, enquanto o Shell permite que você execute programas.
- B O explorador de arquivos é construído no topo do Shell.
- C O Shell é parte do sistema operacional, enquanto o explorador de arquivos é separado.
- D Ambas são interfaces para emitir comandos ao sistema operacional.

## 2 Manipulando arquivos e diretórios

O sistema de arquivos gerencia arquivos e diretórios (ou pastas). Cada um é identificado por um **caminho absoluto** que mostra como acessá-lo a partir do diretório raiz do sistema de arquivos: `/home/meu-usuario` é o diretório `meu-usuario` no diretório `home`, enquanto `/home/meu-usuario/course.txt` é um arquivo `course.txt` nesse diretório, e `/` é o **diretório raiz**, ou seja a origem de todo o sistema de arquivos (ele não é subdiretório de nenhum outro diretório).

### Importante

Ao longo do texto iremos nos referir a `meu-usuario` como sendo o nome de usuário que você utiliza em seu sistema. Assim, sempre iremos nos referir ao seu diretório inicial como `/home/meu-usuario`. Quando for utilizar os nomes de caminhos em seu ambiente, substitua qualquer referência de `meu-usuario` pelo nome de usuário que você configurou em seu sistema.

Como exemplo, se eu configurei meu nome de usuário como `lauro`, então quando aparecer um caminho como `/home/meu-usuario/course.txt`, devo alterar para `/home/lauro/course.txt`.

## Importante

Alguns exemplos e exercícios presentes neste texto utilizam pastas e arquivos específicos estão disponíveis no repositório da disciplina. O link de acesso ao repositório é: <https://github.com/bioinfo/nupeb/nup715>. Para configurar o ambiente, siga os seguintes passos:

1. Abra o seu terminal de comandos.
2. Utilize o comando `cd ~` para acessar o seu diretório inicial de usuário.
3. Agora utilize o seguinte comando para baixar o *script* que irá baixar os demais arquivos e diretórios:
  - `wget https://tinyurl.com/2dfyhwhph -O prepare-workspace.sh`
4. Agora execute o comando `bash -i prepare-workspace.sh` para executar o *script* e realizar os *downloads*.
5. Ao usar o comando `ls`, será possível listar todas as novas pastas baixadas.

## 2.1 Onde estou?

Para descobrir onde você está no sistema de arquivos, execute o comando `pwd` (abreviação de *print working directory* - "imprimir diretório de trabalho"). Isso imprime o caminho absoluto de seu diretório de trabalho atual, que é onde o Shell executa comandos e procura por arquivos por padrão.

### Atividade

Execute o `pwd`. Onde você está agora?



## 2.2 Como posso identificar arquivos e diretórios?

`pwd` diz onde você está. Para descobrir o que está lá, digite `ls` (que é a abreviação de *listing* - "listagem") e pressione a tecla *Enter*. Por padrão, `ls` lista o conteúdo de seu diretório atual (aquele exibido por `pwd`). Se você adicionar os nomes de alguns arquivos, `ls` irá listá-los, e se você adicionar os nomes de diretórios, ele irá listar seus conteúdos. Por exemplo, `ls /home/meu-usuario` mostra o que está em seu **diretório inicial** (geralmente chamado de diretório *home*).

### Dica

Outra forma de listar o que está em seu diretório *home* é através de `ls ~`. Neste comando, o caractere til (~) representa o caminho para o seu diretório inicial.

### Dica

Quando estiver escrevendo no console de comandos, utilize a tecla *Tab* para auto-completar comandos e caminhos. Isto facilitará a sua vida, pois irá acelerar a escrita, evitará erros de digitação e permitirá descobrir novas opções.

O diretório raiz contém todos os arquivos necessários para o funcionamento do sistema, como arquivos de inicialização, bibliotecas, pacotes, binários essenciais, configuração do sistema, arquivos do usuário e arquivos temporários.

## Pergunta

Use `ls` com um argumento apropriado para listar os arquivos no diretório raiz (que contém todos os diretórios). Qual desses diretórios não está listado?

- A home
- B bin
- C linux
- D etc

## 2.3 Como dar nomes aos meus arquivos?

Os sistemas Linux diferenciam as letras maiúsculas de minúsculas, por isso são chamados de *case sensitive*. Os sistemas Windows não fazem essa diferenciação, portanto, se você estiver movendo seus arquivos de uma plataforma de desenvolvimento do Windows para uma plataforma do Linux, pode ser necessário tomar alguns cuidados com os nomes de pastas e arquivos.

Um erro comum é nomear arquivos com uma mistura de maiúsculas e minúsculas. Como em `MeuArquivo.TXT`. Linux lê o nome do arquivo exatamente como ele foi digitado, então `MeuArquivo.TXT` não é o mesmo que `meuarquivo.txt`.

Outra fonte de confusões pode ser em relação ao uso de extensões de arquivos. Muitas vezes, nos sistemas Windows, não é necessário verificar a extensão de um arquivo. Entretanto, no sistemas Linux, devemos ficar atentos. As extensões de arquivos também são **case sen-**

**sitive.** Elas podem variar no número de caracteres, geralmente 3 ou 4, mas não é restrito a estes números. No sistema Linux, é possível criar arquivos sem extensão, inclusive.

Algumas extensões comuns de arquivos com três caracteres são: *txt*, *csv*, *doc*, *mp3*, *png*, *jpg*, etc. Com quatro caracteres temos: *jpeg*, *html*, *docx*, *xlsx*, etc.

Note que é possível a um mesmo formato possuir mais de uma forma nomenclatura de extensão. Como é o caso de *jpg* e *jpeg*, ambos são relativos ao mesmo formato de arquivo, o JPEG ("Joint Photographic Experts Group"), um formato de imagem padrão que possui dados de imagem compactados com poucas perdas. No entanto, para o sistema Linux, a extensão *jpg* não é a mesma que *jpeg*, pois não são escritas estritamente iguais, isso pode levar a alguns problemas ou inconvenientes. É importante que um padrão seja adotado para evitar dores de cabeça, ou seja, escolha uma grafia única para as extensões de um determinado formato de arquivo.

Algumas extensões de arquivos comuns na bioinformática são: *fasta*, *fastq*, *bam*, *sam*, *vcf*, *gff*, dentre outras. Lembre-se que podem haver mais de uma forma de escrita de uma extensão. A formato de arquivo FASTA possui diversas possibilidades extensão, alguns exemplos são: *fasta*, *fna*, *ffn*, *faa*, *frn* e *fa*.

Adotar algumas boas práticas de nomenclatura de pastas e arquivos podem ajudar quando estiver usando o *Shell* para manipulá-los. Evite usar caracteres de espaço, acentos. para dar nomes a arquivos ou pastas. Por exemplo, ao invés de usar `meu arquivo.txt`, nomeie seu ar-

quivo como `meu-arquivo.txt`. Outro exemplo seria ao invés de usar um nome de diretório como `meu diretorio.txt`, utilize um nome sem espaços e sem acentos, como `meu-diretorio.txt`

Abaixo são sumarizadas algumas das boas práticas para definição de nomes de arquivos e pastas nos sistemas Linux.

### Dica

Para evitar problemas com os caminhos de arquivo, apresentamos a seguir as boas práticas recomendadas para nomenclatura de arquivos. Veja abaixo:

1. Nomeie todos os seus arquivos em letras minúsculas.
2. Em vez de usar um espaço, use um sublinhado (`_`) ou um hífen (`-`).
3. Somente caracteres alfanuméricos (evite letras com acentos ou cedilha), pontos, não use símbolos como `%`, `$` e assim por diante. Reserve sublinhados e hífens para substituir os espaços.
4. Use tipos de arquivo consistentes. Use *fasta* ou *fna*, por exemplo. Não use os dois.
5. Mantenha os nomes dos arquivos curtos e descritivos.

## 2.4 De que outra forma posso identificar meus arquivos e diretórios?

Um **caminho absoluto** (*absolute path*) detalha completamente o caminho de um arquivo ou pasta. Ele parte do elemento raiz e segue por todos os subdiretórios até onde está o arquivo ou pasta que você deseja referenciar. É como uma coordenada de latitude e longitude, ou seja, tem o mesmo valor, não importa onde você esteja no sistema. Um

**caminho relativo** (*relative path*), por outro lado, especifica um determinado local a partir de onde você está no momento. Seria como dizer que um dado ponto está a “20 quilômetros ao norte”, ao invés de dar a coordenada exata daquele local. Alguns exemplos:

- Se você estiver no diretório `/home/meu-usuario`, o caminho relativo `data` especifica o mesmo diretório que o caminho absoluto `/home/meu-usuario/data`.
- Se você estiver no diretório `/home/meu-usuario/data`, o caminho relativo `fasta-files/fasta-sequences-example.faa` especifica o mesmo arquivo que o caminho absoluto `/home/meu-usuario/data/fasta-files/fasta-sequences-example.faa`.

O Shell decide se um caminho é absoluto ou relativo observando seu primeiro caractere: Se começar com o caractere barra (`/`), é absoluto. Se não começar com `/`, é relativo.

### Atividade

1. Utilize o comando `cd ~` para se posicionar no diretório inicial.
2. Utilize `ls` com um **caminho relativo** para listar o conteúdo de `/home/meu-usuario/data/fasta-files`
3. Ainda no seu diretório inicial, utilize `ls` com um **caminho relativo** para listar apenas o arquivo `/home/meu-usuario/data/fasta-files/fasta-sequences-example.faa`.

## 2.5 Como posso mover para outro diretório?

Assim como você pode se mover em um navegador de arquivos clicando duas vezes nas pastas, você pode se mover no sistema de arquivos usando o comando `cd` (que significa *change directory* - "alterar diretório").

Se você digitar `cd data` e, em seguida, digitar `pwd`, o Shell lhe dirá que agora você está em `/home/meu-usuario/data`. Se você executar o `ls` sozinho, ele mostrará o conteúdo de `/home/meu-usuario/data`, porque é onde você está. Se você quiser voltar ao seu diretório *home* `/home/meu-usuario`, você pode usar o comando `cd /home/meu-usuario` ou `cd ~`.

### Atividade

1. Utilize o comando `cd ~` para se posicionar no diretório inicial.
2. Mude para o diretório `data` utilizando um **caminho relativo**.
3. Use o comando `pwd` para checar onde você está.
4. Utilize `ls` sem informar nenhum caminho para listar o que está em seu diretório corrente.

## 2.6 Como posso subir um diretório?

O "pai" de um diretório é o diretório acima dele na hierarquia da estrutura de diretórios de seu sistema. Por exemplo, `/home` é o pai de `/home/meu-usuario` e `/home/meu-usuario` é o pai de `/home/meu-usuario/data`.

Você sempre pode fornecer o caminho absoluto de seu diretório pai para comandos como `cd` e `ls`. Com mais frequência, porém, você aproveitará o fato de que o caminho especial `..` (dois pontos sem espaços entre eles) significa "o diretório acima daquele em que estou atualmente".

Se você estiver em `/home/meu-usuario/data`, ao usar o comando `cd ..` você será movido para `/home/meu-usuario`. Se você usar `cd ..` mais uma vez, ele o coloca em `/home`. Mais um `cd ..` coloca você no diretório raiz `/`, que é o topo do sistema de arquivos. (Lembre-se de colocar um espaço entre `cd` e `..`, pois é um *comando* e um *caminho*, não um único comando de quatro letras.)

Um único ponto, `.`, sempre significa "o diretório atual", então `ls` apenas e `ls .` fazem a mesma coisa, enquanto `cd .` não tem efeito (porque o move para o diretório em que já está atualmente).

Lembrando que um caminho especial é `~` (o caractere til), que significa "seu diretório pessoal inicial", como `/home/meu-usuario`. Não importa onde você esteja, `ls ~` sempre listará o conteúdo de seu diretório pessoal e `cd ~` sempre o levará para a sua pasta inicial, a sua *home*.

### Pergunta

Se você estiver em `/home/meu-usuario/data`, onde `cd ~/../.` irá levá-lo?

- A `/home/meu-usuario`
- B `/home`
- C `/home/meu-usuario`
- D `/` (o diretório raiz - *root*)

## 2.7 Como posso copiar arquivos?

Freqüentemente, você desejará copiar arquivos, movê-los para outros diretórios para organizá-los ou renomeá-los. Um comando para fazer isso é `cp`, que é a abreviação de *copy* - "copiar". Se, por exemplo, `original.txt` for um arquivo existente, então:

### Copiando um arquivo na mesma pasta

```
1 cp original.txt duplicate.txt
```

cria uma cópia de `original.txt` chamada `duplicate.txt`. Se já houver um arquivo chamado `duplicate.txt`, ele será sobrescrito. Se o último parâmetro para `cp` for um diretório existente, um comando como:

### Copiando um arquivo para uma pasta diferente

```
1 cp data/csv/grades.csv backup
```

copia o arquivo `grades.csv` para o diretório `backup`. Você pode também especificar mais de uma coisa para ser copiada para um mesmo destino. Como em:

### Copiando mais de um arquivo para uma pasta

```
1 cp data/csv/biostats.csv data/csv/grades.csv backup
```

que irá copiar o arquivo `biostats.csv` e o arquivo `grades.csv` para o diretório `backup`. Ou seja, quando usamos o comando `cp`, o último ca-



minho informado define o local de destino da cópia, enquanto todos os outros caminhos anteriores definem os conteúdos que serão copiados para este destino.

### Atividade

1. Utilize o comando `cd ~` para se posicionar no diretório inicial.
2. Faça uma cópia do arquivo `data/csv/biostats.csv` para o diretório `backup`, que está na pasta inicial, renomeando-o para `biostats.bkp`.
3. Sem sair da pasta inicial, copie os arquivos `biostats.csv` e `grades.csv` da pasta `data/csv` para o diretório `backup`.

## 2.8 Como posso mover um arquivo?

Enquanto o `cp` copia um arquivo, o comando `mv` (*move* - “mover”) o move de um diretório para outro, como se você o tivesse arrastado em um navegador gráfico de arquivos. Ele lida com seus parâmetros da mesma maneira que `cp`. Portanto, se você estiver no diretório `data/csv`, o comando:

### Movendo arquivos para o diretório pai

```
1 mv biostats.csv grades.csv ..
```

irá mover os arquivos `biostats.csv` e `grades.csv` do diretório atual um nível acima, para o seu diretório pai (pois, como já vimos, `..` se refere sempre ao diretório diretamente acima de sua localização atual).

## Atividade

1. Utilize o comando `cd ~` para se posicionar no diretório inicial.
2. Utilizando um único comando, mova os arquivos `biostats-copia.csv` e `grades-copia.csv` da pasta `data/csv` para o diretório `backup`.

## 2.9 Como posso renomear arquivos?

O comando `mv` também pode ser usado para renomear arquivos. Considere que você está no diretório `data/csv`. Se você executar:

### Movendo arquivos para o diretório pai

```
1 mv biostats.csv renamed-biostats.csv
```

então o arquivo `biostats.csv` no diretório de trabalho atual é “movido” para o arquivo `renamed-biostats.csv`. Isso é um pouco diferente da maneira como os navegadores de arquivos com interfaces gráficas funcionam, mas geralmente é útil.

Um aviso: assim como o `cp`, o `mv` substituirá os arquivos existentes. Se, por exemplo, você já tem um arquivo chamado `renamed-biostats.csv`, o comando mostrado acima irá substituí-lo pelo que estiver em `biostats.csv`.

## Atividade

1. Utilize o comando `cd ~` para se posicionar no diretório inicial.
2. Mova para a pasta `data/csv`.
3. Faça uma cópia do arquivo `biostats.csv` com o nome `novo-biostats.csv` na mesma pasta.
4. Faça uma cópia do arquivo `grades.csv` com o nome `novo-grades.csv` na mesma pasta.
5. Renomeie o arquivo `novo-biostats.csv` para `novo-biostats.csv.bkp`.
6. Renomeie o arquivo `novo-grades.csv` para `novo-grades.csv.bkp`.
7. Liste o conteúdo da pasta com o comando `ls` para ver se está tudo correto.

## 2.10 Como posso deletar arquivos?

Podemos copiar arquivos e movê-los; para excluí-los, usamos `rm`, que significa *remove* - "remover". Assim como com `cp` e `mv`, você pode dar a `rm` os nomes de quantos arquivos desejar. Suponha que hajam os arquivos `thesis.txt` e `backup/thesis-2022-10.txt`, o comando:

### Movendo arquivos para o diretório pai

```
1 rm thesis.txt backup/thesis-2022-08.txt
```

removerá os arquivos `thesis.txt` e `backup/thesis-2022-10.txt`.

`rm` faz exatamente o que seu nome diz, e faz isso de imediato: ao contrário de navegadores de arquivos gráficos, o Shell não tem uma pasta "Lixeira", então quando você digita o comando acima, seus arquivos de-

saparecem para sempre. Então, muito cuidado ao utilizá-lo.

### Atividade

1. Utilize o comando `cd ~` para se posicionar no diretório inicial.
2. Mova para a pasta `data/csv`.
3. Remova o arquivo `novo-biostats.csv.bkp`.
4. Retorne para o seu diretório inicial (utilize o `cd`).
5. Remova, sem mudar de diretório, o arquivo `novo-grades.csv.bkp` que está na pasta `data/csv`.

## 2.11 Como posso criar e excluir diretórios?

`mv` trata os diretórios da mesma forma que trata os arquivos: se você estiver em seu diretório inicial e executar `mv data my-data`, por exemplo, `mv` muda o nome do diretório `data` para `my-data`. No entanto, `rm` funciona de maneira diferente.

Se você tentar `rm` em um diretório, o Shell exibirá uma mensagem de erro informando que não é possível fazer isso, principalmente para impedi-lo de excluir acidentalmente um diretório inteiro cheio de trabalho. Em vez disso, você pode usar um comando separado chamado `rmdir`. Para maior segurança, ele só funciona quando o diretório está vazio, portanto, você deve excluir os arquivos de um diretório antes de excluí-lo. (Usuários experientes podem usar a opção `-r` para `rm` para obter o mesmo efeito; discutiremos as opções de comando mais à frente.)

### Atividade

1. Se posicione no diretório inicial.
2. Utilizando o comando `rmdir`, tente apagar a pasta `useless-folder`.
3. Sem sair da pasta inicial, apague o arquivo `useless-file.txt` no diretório `useless-folder`.
4. Agora que a pasta `useless-folder` está vazia, tente apagá-la novamente.

Para criar novos diretórios, utilizamos o comando `mkdir` que significa *make directory* - "criar diretório". Basta informar o nome da nova pasta para o comando. Assim, `mkdir nome-da-pasta` irá criar um novo diretório (vazio) chamado `nome-da-pasta` que terá como diretório pai o diretório corrente.

### Atividade

1. Se posicione no diretório inicial.
2. Use o comando `mkdir` para criar um novo diretório chamado `dados-anuais` na sua pasta inicial.
3. Agora crie uma nova pasta chamada `dados-2022` dentro da pasta `dados-anuais`, mas sem sair da sua pasta inicial.

## 2.12 Como criar novos arquivos?

Para criar novos arquivos, podemos utilizar o comando `touch`. Basta informarmos o nome do arquivo a ser criado. É possível criar mais de um arquivo com um único comando, basta informar todos os nomes de

uma só vez. Assim, `touch file-01.txt file-02.txt file-03.csv` irá criar os arquivos `file-01.txt`, `file-02.txt` e `file-03.txt` no diretório corrente. Note também que é possível criar arquivos com qualquer tipo de extensão. Eles serão criados vazios, ou seja, sem conteúdo algum.

### Atividade

1. Se posicione no diretório inicial.
2. Use o comando `touch`, sem sair da sua pasta inicial, para criar os arquivos `tabela-2022.csv` e `relatorio-2022.txt` na pasta `dados-anuais/dados-2022`. Utilize caminhos relativos.

## 2.13 Juntando tudo

Freqüentemente, você criará arquivos intermediários ao analisar dados. Em vez de armazená-los em seu diretório pessoal, você pode colocá-los em `/tmp`, que é onde as pessoas e os programas geralmente mantêm os arquivos temporários, ou seja, de que precisam apenas brevemente. Observe que `/tmp` está imediatamente abaixo do diretório raiz `/` (é filho do diretório raiz), não abaixo do seu diretório inicial (`/home/meu-usuario`). Este exercício mostrará como fazer isso.

## Atividade

1. Se posicione no diretório `/tmp`.
2. Liste o conteúdo `/tmp` sem escrever um nome de diretório.
3. Crie uma nova pasta chamada `rascunho` dentro de `/tmp`.
4. Faça uma cópia do arquivo `/home/meu-usuario/data/csv/biostats.csv` para dentro de `/tmp/rascunho`. Utilize o atalho `cp` para o nome do diretório inicial ao especificar o caminho do arquivo e use um caminho relativo para a pasta `rascunho` ao invés do caminho absoluto.
5. Agora crie um novo arquivo chamado dentro de `meu-texto.txt` dentro de `/tmp/rascunho`, sem sair da pasta `/tmp`. Utilize um caminho relativo.
6. Liste o conteúdo do diretório `/tmp/rascunho` sem sair da pasta `/tmp`.

## 3 Manipulando dados

Os comandos que você viu no capítulo anterior permitiram que você movesse as coisas no sistema de arquivos. Este capítulo mostrará como trabalhar com os dados desses arquivos. As ferramentas que usaremos são bastante simples, mas são blocos de construção sólidos, necessários para entender o que podemos fazer.

### 3.1 Como posso ver o conteúdo de um arquivo?

Antes de renomear ou excluir arquivos, você pode querer dar uma olhada em seu conteúdo. A maneira mais simples de fazer isso é com o comando `cat`, que apenas imprime o conteúdo dos arquivos na tela. (Seu nome é uma abreviação de *concatenate* (“concatenar”), que significa “ligar as coisas”, uma vez que imprimirá todos os arquivos cujos nomes você der, um após o outro). Para imprimir o conteúdo de arquivos basta informar os caminhos para o comando. Assim, `cat meu-arquivo.txt` imprime o conteúdo do arquivo `meu-arquivo.txt` na tela do terminal de comandos.

#### Atividade

1. Se posicione no diretório inicial.
2. Imprima na tela o conteúdo do arquivo `fasta-sequences-example.faa`, que está na pasta `data/fasto-files`, sem sair do diretório inicial.
3. Utilizando apenas um comando, imprima os conteúdos dos arquivos `biostats.csv` e `grades.csv`, que estão na pasta `data/csv`, sem sair do diretório inicial.



### 3.2 Como posso ver o conteúdo de um arquivo pedaço por pedaço?

Você pode usar `cat` para imprimir arquivos grandes e, em seguida, rolar pela saída, mas geralmente é mais conveniente paginar a saída. O comando original para fazer isso era chamado `more`, mas foi substituído por um comando mais poderoso chamado `less`. (Esse tipo de nomenclatura é uma forma de humor no mundo Unix.) Quando você utiliza o comando `less` em um arquivo, uma página é exibida por vez; você pode pressionar a barra de espaço para descer a página ou digitar `q` para sair.

Se você der os nomes de vários arquivos para o comando `less`, você poderá digitar `:n` (dois pontos e "n" minúsculo) para mover para o próximo arquivo (*next* - "próximo"), `:p` (*previous* - "anterior") para voltar ao anterior, ou `:q` (*quit* - "sair") para sair. Você pode usar a barra de espaço para descer na página, bem como as setas direcionais. Utilizando a tecla "Home", você irá diretamente para o topo da página, enquanto com a tecla "End", você vai direto para o fim da página.

## Atividade

1. Se posicione no diretório inicial.
2. Vá até a pasta `data/txt-files`.
3. Utilize o comando `less` e informe o nome dos arquivos `texto-01.txt` e `texto-02.txt`, nesta ordem, para visualizá-los simultaneamente.
4. Explore a página de texto com as setas direcionais, com a barra de espaço e com as teclas "Home" e "End".
5. Utilize `:n` para ir para a próxima página (arquivo) e depois retorne com `:p`. Por fim, use `:q` para sair e retornar à linha de comandos.

### 3.3 Como posso ver o início de um arquivo?

A primeira coisa que a maioria dos bioinformatas faz quando recebe um novo conjunto de dados para analisar é descobrir quais campos ele contém e quais valores esses campos têm. Se o conjunto de dados foi exportado de um banco de dados ou planilha, geralmente será armazenado como valores separados por vírgula (CSV). Uma maneira rápida de descobrir o que ele contém é examinar as primeiras linhas.

Podemos fazer isso no Shell usando um comando chamado `head`. O comando de nome *head* ("cabeça") imprime as primeiras linhas de um arquivo (por padrão, as 10 primeiras linhas).

## Atividade

1. Se posicione no diretório inicial.
2. Imprima as primeiras linhas do arquivo `biostats.csv` da pasta `data/csv`.

## Pergunta

O que o comando `head` faz se não houver 10 linhas no arquivo? Descubra, usando-o para visualizar as linhas iniciais do arquivo `data/`.

- A Imprime uma mensagem de erro porque o arquivo é muito curto.
- B Exibe apenas as linhas presentes no arquivo.
- C Exibe linhas em branco suficientes para aumentar o total para 10.

### 3.4 Como posso controlar o que os comandos fazem?

Você nem sempre vai querer olhar as primeiras 10 linhas de um arquivo, então o Shell permite que você mude o comportamento do comando `head` dando a ele um sinalizador de linha de comando chamado *flag*. Se você executar o comando:

#### Movendo arquivos para o diretório pai

```
1 head -n 3 data/csv/grades.csv
```

`head` exibirá apenas as três primeiras linhas do arquivo. Se você executar `head -n 100`, ele exibirá as primeiras 100 (assumindo que haja tantos) e assim por diante.

O nome de um sinalizador geralmente indica seu propósito (por exemplo, `-n` significa "número de linhas"). As *flags* de comando não precisam ser um caractere `-` seguido por uma única letra, mas é uma convenção amplamente usada.

Observação: é considerado uma boa prática colocar todas as *flags* antes de quaisquer nomes de arquivo.

### Atividade

1. Imprima apenas a 5 primeiras linhas do arquivo `biostats.csv` que está na pasta `data/csv`.

## 3.5 Como posso listar tudo abaixo de um diretório?

Para ver tudo sob um diretório, não importa o quão profundamente aninhado ele esteja, você pode dar a `ls` o sinalizador `-R` (que significa "recursivo"). Se você usar `ls -R` em seu diretório inicial, verá todos os arquivos e diretórios no nível atual, depois tudo em cada subdiretório e assim por diante.

Para ajudá-lo a saber o que é cada item listado, `ls` tem outro sinalizador `-F` que imprime uma `/` após o nome de cada diretório e um `*` após o nome de cada programa executável.

### Atividade

1. Execute `ls` com os dois sinalizadores, `-R` e `-F`, e o caminho absoluto para seu diretório inicial para ver tudo o que ele contém. (A ordem dos sinalizadores não importa, mas o nome do diretório deve vir por último.)

## 3.6 Como posso obter ajuda para um comando?

Para descobrir o que os comandos fazem, as pessoas costumavam usar o comando `man` (abreviação de “manual”). Por exemplo, o comando `man head` traz esta informação:

```
Execução do comando man head

1 HEAD(1)                BSD General Commands Manual                HEAD(1)
2
3 NAME
4     head -- display first lines of a file
5
6 SYNOPSIS
7     head [-n count | -c bytes] [file ...]
8
9 DESCRIPTION
10    This filter displays the first count lines or bytes of each of
11    the specified files, or of the standard input if no files are
12    specified. If count is omitted it defaults to 10.
13
14    If more than a single file is specified, each file is preceded by
15    a header consisting of the string ``==> XXX <=='' where ``XXX''
16    is the name of the file.
17
18 SEE ALSO
19     tail(1)
```

`man` invoca `less` automaticamente, então você pode precisar pressionar a barra de espaço para percorrer as informações e `:q` para sair.

A descrição de uma linha em `NOME` informa brevemente o que o comando faz, e o resumo em `SINOPSE` lista todos as *flags* que ele entende. Tudo o que é opcional é mostrado entre colchetes `[...]`, as alternativas são separadas por `|`, e as coisas que podem ser repetidas

são mostradas por `...`, então a página do manual do comando `head` está dizendo que você pode fornecer um contagem de linha com `-n` ou contagem de byte com `-c`, e você pode atribuir a ela qualquer número de nomes de arquivo.

### Atividade

1. Leia a página do manual do comando `tail` para descobrir o que fazer com um sinal `+` antes do número usado com a flag `-n`. (Lembre-se de pressionar as setas direcionais e a barra de espaço para mover-se pela página e digitar `q` para sair.)
2. Use `tail` com a flag `-n +7` para exibir tudo, exceto as seis primeiras linhas do arquivo `data/csv/biostats.csv` (i.e. a impressão começa pela sétima linha). Lembre-se que o nome do arquivo vem por último.

## 3.7 Como posso selecionar colunas de um arquivo?

`head` e `tail` permitem que você selecione linhas de um arquivo de texto. Se você deseja selecionar colunas, você pode usar o comando `cut`. Possui várias opções (use `man cut` para explorá-las), mas uma forma comum de utilizá-lo é algo como:

### Exemplo de uso do comando `cut`

```
1 cut -f 2-5,8 -d , values.csv
```

que significa "selecione as colunas 2 a 5 e a coluna 8, usando a vírgula como separador". `cut` usa `-f` (que significa "campos") para especificar colunas e `-d` (que significa "delimitador") para especificar o separador de colunas. Você precisa especificar o último porque alguns arquivos

podem usar espaços, tabulações ou dois-pontos para separar colunas.

### Pergunta

Qual comando selecionará a primeira coluna (contendo os nomes) do arquivo `data/csv/biostats.csv`?

- A `cut -d , -f 1 data/csv/biostats.csv`
- B `cut -d, -f1 data/csv/biostats.csv`
- C Qualquer uma das opções acima.
- D Nenhuma das opções acima, porque `-f` deve vir antes de `-d`.

## 3.8 O que cut não pode fazer?

`cut` é um comando simplório. Em particular, ele não entende *strings* (sequência de caracteres) entre aspas. Se, por exemplo, seu arquivo for:

```
1 Name, Age
2 "Johel, Ranjit", 28
3 "Sharma, Rupinder", 26
```

então:

```
1 cut -f 2 -d , everyone.csv
```

produzirá:

```
1 Age
2 Ranjit"
3 Rupinder"
```

em vez da idade de todos, porque ele pensará que a vírgula entre o sobrenome e o nome é um separador de colunas.

### Pergunta

Qual é a saída do comando `cut -d: -f 2-4` ao processar a linha: `data/csv/biostats.csv` ?

```
1 first:second:third:
```

(Observe os dois pontos)

- A `second`
- B `second:third`
- C `second:third:`
- D Nenhuma das opções acima, porque não existem quatro campos.

## 3.9 Como posso repetir comandos?

Uma das maiores vantagens de usar o Shell é que ele torna mais fácil fazer as coisas novamente. Se você executar alguns comandos, poderá pressionar a tecla de seta para cima para percorrê-los de volta. Você também pode usar as teclas de seta esquerda e direita e a tecla *Delete* para editá-los. Pressionar *Enter* executará o comando modificado.

O comando `history` ("histórico") imprimirá uma lista de comandos executados recentemente. Cada um é precedido por um número de série



para facilitar a reexecução de comandos específicos: basta digitar `!55` para reexecutar o 55º comando em seu histórico (se você tiver tantos). Você também pode executar novamente um comando digitando um ponto de exclamação seguido do nome do comando, como `!head` ou `!cut`, que executará novamente o uso mais recente desse comando.

### Atividade

1. Execute `head biostats.csv` em seu diretório inicial (que deve falhar).
2. Mude para o diretório `data/csv`.
3. Reexecute o comando `head` com `!head`.
4. Use o comando `history` para ver o que você fez.
5. Reexecute o comando `head` novamente usando o `!` seguido por um número do histórico de comando.

## 3.10 Como posso selecionar linhas contendo valores específicos?

`head` e `tail` selecionam linhas, `cut` seleciona colunas e `grep` seleciona linhas de acordo com o que elas contêm. Em sua forma mais simples, `grep` pega um pedaço de texto seguido por um ou mais nomes de arquivo e imprime todas as linhas dos arquivos que contêm esse texto. Por exemplo, `grep \"M\" data/csv/biostats.csv` imprime linhas do arquivo `biostats.csv` que contêm a correspondência `'M'`, ou seja, as linhas onde a coluna `'Sex'` possui o valor `"M"` de *Male*.

Note que utilizamos a barra invertida `\` antes das aspas duplas `"` ao

escrevemos `\ "M\"`. Isto porque alguns caracteres possuem funções especiais na linha de comando e aspas duplas é um deles. Por isso devemos utilizar a barra invertida antes do caractere especial, pois `\` é o caracteres especial de *escape*, i.e., ele diz ao Shell para interpretar o próximo caractere de forma literal e não por sua função especial.

Algumas das *flags* mais comuns do `grep` são:

- `-c`: imprime uma contagem de linhas correspondentes em vez das próprias linhas
- `-h`: não imprime os nomes dos arquivos ao pesquisar vários arquivos
- `-i`: não distingue entre maiúsculas e minúsculas (por exemplo, trata "Regressão" e "regressão" como correspondências).
- `-l`: imprime os nomes dos arquivos que contêm as correspondências, não as correspondências em si.
- `-n`: imprime números de linhas onde houve correspondências encontradas.
- `-v`: inverte a correspondência, ou seja, mostra apenas as linhas que não possuem correspondências.

## Atividade

1. Imprima o conteúdo de todas as linhas que contêm correspondências com a *string* `\>` no arquivo `data/fasto-files/fasto-sequences-example.faa` executando um único comando no diretório inicial. Não use *flags*. Em arquivos do formato FASTA, linhas que começam com o caractere `>` (maior que) contêm os identificadores e as informações relativas às sequências, ou seja, identifica estas linhas. Note que você deve utilizar a barra invertida para informar o *escape* do caractere especial `>` ao Shell, assim, ele irá considerar o seu valor literal, ao invés de sua função especial na linha de comando.
2. Inverta a correspondência para encontrar todas as linhas que não contêm a *string* `\>` em `data/fasto-files/fasto-sequences-example.faa` e mostre os seus números de linha correspondentes. Lembre-se de que é considerado uma boa prática colocar todas as *flags* antes de outros valores, como nomes de arquivos ou o termo de pesquisa `\>`.
3. Conte quantas linhas contêm a *string* `\>` nos arquivos `data/fasto-files/fasto-sequences-example.faa` e `data/fasto-files/ecoli_promoters.faa` combinados. (Novamente, execute um único comando de seu diretório inicial.)

### 3.11 Por que nem sempre é seguro tratar os dados como texto?

A seção `SEE ALSO` da página do manual para `cut` refere-se a um comando chamado `paste` que pode ser usado para combinar arquivos de dados em vez de cortá-los.

## Pergunta

Leia a página de manual para `paste` e, em seguida, execute `paste` para combinar os dados dos arquivos `data/csv/biostats.csv` e `data/csv/grades.csv` em uma única tabela usando uma vírgula como separador. O que há de errado com a saída do ponto de vista da análise de dados?

- A Os cabeçalhos das colunas são repetidos.
- B As últimas linhas têm o número errado de colunas.
- C Alguns dados de `biostats.csv` estão faltando.

## 4 Combinando ferramentas

O verdadeiro poder do Shell Unix não está nos comandos individuais, mas na facilidade com que eles podem ser combinados para fazer coisas novas. Este capítulo mostrará como usar esse poder para selecionar os dados desejados e introduzirá comandos para classificar valores e remover duplicatas.

### 4.1 Como armazenar a saída de um comando em um arquivo?

Todas as ferramentas que você viu até agora permitem especificar os nomes de arquivos de entrada. A maioria não tem a opção de nomear um arquivo de saída porque não precisa de um. Em vez disso, você pode usar o **redirecionamento** para salvar a saída de qualquer comando em qualquer lugar que desejar. Se você executar este comando:

```
1 head -n 3 data/csv/biostats.csv
```

ele irá imprimir as 3 primeiras linhas dos dados do arquivo `biostats.csv` na tela. Se você executar este comando em vez disso:

```
1 head -n 3 data/csv/biostats.csv > data/csv/top-biostats.csv
```

nada irá aparecer na tela. Em vez disso, a saída de `head` é colocada em um novo arquivo chamado `data/csv/top-biostats.csv`. Você pode dar uma olhada no conteúdo desse arquivo usando `cat`:

```
1 cat data/csv/top-biostats.csv
```

O caractere especial maior que `>` diz ao Shell para redirecionar a saída do comando `head` para um arquivo. Não faz parte do comando principal; em vez disso, funciona com todos os comandos do Shell que produzem saída. Por este motivo utilizamos o caractere de *escape* na Seção 3.10, para que o comando `grep` não confunda o valor literal do sinal `>` com sua função especial de redirecionamento de saída.

### Atividade

1. Combine o comando `tail` com redirecionamento para salvar as últimas 5 linhas de `data/csv/grades.csv` em um arquivo chamado `data/csv/last-grades.csv`.

## 4.2 Como posso usar a saída de um comando como entrada de outro?

Suponha que você deseja obter as linhas do meio de um arquivo. Mais especificamente, suponha que você queira obter as linhas 3 a 5 de um de nossos arquivos de dados. Você pode começar usando `head` para obter as primeiras 5 linhas e redirecionar para um arquivo, e então usar `tail` para selecionar as últimas 3:

```
1 head -n 5 data/csv/grades.csv > data/csv/top-grades.csv
2 tail -n 3 data/csv/top-grades.csv
```

Uma verificação rápida confirma que se trata das linhas 3 a 5 de nosso

arquivo original, porque são as últimas 3 linhas das 5 primeiras.

### Atividade

1. Selecione as duas últimas linhas de season / `data/csv/grades.csv` e salve-as em um arquivo chamado `data/csv/bottom-grades.csv`.
2. Selecione a primeira linha de `data/csv/bottom-grades.csv` para obter a penúltima linha do arquivo original.

## 4.3 Qual é a melhor maneira de combinar comandos?

Usar o redirecionamento para combinar comandos tem duas desvantagens:

1. Ele deixa muitos arquivos intermediários espalhados (como `top-grades.csv`).
2. Os comandos para produzir seu resultado final estão espalhados por várias linhas do histórico.

O Shell fornece outra ferramenta que resolve esses dois problemas ao mesmo tempo, chamada de **pipe**. Mais uma vez, comece executando o `head`:

```
1 head -n 5 data/csv/biostats.csv
```

Em vez de enviar a saída de `head` para um arquivo, adicione uma barra vertical `|` (o caractere *pipe*) e o comando `tail` sem um nome de arquivo:

```
1 head -n 5 data/csv/biostats.csv | tail -n 3
```

O símbolo de *pipe* diz ao Shell para usar a saída do comando à esquerda como entrada para o comando à direita.

### Atividade

1. Use `cut` para selecionar todos os nomes da coluna 1 do arquivo delimitado por vírgulas `data/csv/biostats.csv`, em seguida, canalize o resultado (utilize o *pipe*) para o comando `grep`, com uma correspondência invertida, para excluir a linha do cabeçalho que contém a palavra "Name". `cut` e `grep` foram abordados em detalhes anteriormente.

## 4.4 Como posso combinar vários comandos?

Você pode encadear qualquer número de comandos juntos. Por exemplo, este comando:

```
1 cut -d , -f 1 data/csv/biostats.csv | grep -v Name | head -n 3
```

irá:

- selecionar a primeira coluna dos dados do arquivo `biostats.csv`;
- remover a linha do cabeçalho que contém a palavra "Name"; e
- selecionar as primeiras 3 linhas dos dados atuais (aqueles que passaram por todos os filtros anteriores da cadeia de comandos).



## Atividade

Ao encadear vários comandos, você pode criar poderosos *pipelines* de manipulação de dados.

1. No exercício anterior, você usou o seguinte comando para selecionar todos os nomes coluna 1 de `data/csv/biostats.csv`:

```
1 cut -d , -f 1 data/csv/biostats.csv | grep -v Name
```

Estenda este *pipeline* com um comando `tail` para selecionar apenas o último nome.

## 4.5 Como posso contar os registros em um arquivo?

O comando `wc` (abreviação de - *word count* "contagem de palavras") imprime o número de caracteres, palavras e linhas em um arquivo. Você pode fazer com que ele imprima apenas um deles usando as *flags* `-c`, `-w`, ou `-l`, respectivamente.

## Atividade

1. Conte quantas linhas em `data/txt-files/texto-02.txt` têm a palavra "política". Para fazer isso, use `grep` com o termo desejado para selecionar as linhas e canalize este resultado para o comando `wc` com uma *flag* apropriada para contar as linhas.

## 4.6 Como posso especificar muitos arquivos de uma só vez?

A maioria dos comandos do Shell funcionará em vários arquivos se você informar a eles vários nomes de arquivo. Por exemplo, você pode obter a primeira coluna de todos os arquivos tabulares de uma vez, desta forma:

```
1 cut -d , -f 1 data/csv/biostats.csv data/csv/grades.csv
```

Mas digitar os nomes de muitos arquivos repetidamente é uma má ideia: isso é uma perda de tempo e, mais cedo ou mais tarde, você deixará um arquivo de fora ou repetirá o nome de um arquivo. Para tornar sua vida melhor, o Shell permite que você use **curingas** (*wildcards*) para especificar uma lista de arquivos com uma única expressão. O curinga mais comum é `*`, que significa "corresponder a zero ou mais caracteres". Usando-o, podemos encurtar o comando `cut` acima para este:

```
1 cut -d , -f 1 data/csv/*
```

ou especificando a extensão do arquivo, para os casos onde hajam mais de um formato de arquivos na mesma pasta. O exemplo a seguir ilustra isso:

```
1 cut -d , -f 1 data/csv/*.csv
```

Note que `*.csv` diz uma correspondência de zero ou mais caracteres quaisquer (que é a função do caractere curinga `*`) seguido especifica-

mente pela cadeia de caracteres `.csv`, que justamente define a extensão. Ou seja, o comando `cut` será aplicado em todos os arquivos que possuem um nome qualquer terminado com `.csv` na pasta `data/csv`.

### Atividade

Saber utilizar os *wildcards* (curingas) se torna mais importante se o seu diretório contiver centenas ou milhares de arquivos, algo muito comum de acontecer em *pipelines* de processamento de dados de bioinformática.

Vamos selecionar todos os arquivos `.csv` que estão em cada uma das pastas filhas do diretório `data`.

1. Vá para o diretório inicial.
2. Grave um único comando usando o `head` para obter as três primeiras linhas de cada arquivo com extensão `.csv` presentes nos diretórios filhos da pasta `data` mas não dos arquivos de que possuam outra extensão, como `.faa` ou `.txt`, por exemplo. Use caracteres curingas em vez de soletrar os nomes das pastas e dos arquivos por extenso ao definir a entrada do comando.

## 4.7 Que outros curingas posso usar?

O Shell tem outros curingas também, embora sejam menos usados:

1. `?` corresponde a um único caractere, então `201?.txt` corresponderá a `2022.txt` ou `2020.txt`, mas não a `2022-01.txt`.
2. `[...]` corresponde a qualquer um dos caracteres dentro dos colchetes, então `202[02].txt` corresponde a `2020.txt` ou `2022.txt`, mas não a `2025.txt`.
3. `{...}` corresponde a qualquer um dos padrões separados por vírgula dentro das chaves, então `{*.txt, *.csv}` corresponde a

qualquer arquivo cujo nome termine com `.txt` ou `.csv`, mas não arquivos cujos nomes terminem com `.pdf`.

### Pergunta

Qual expressão corresponderia a `singh.pdf` e `johel.txt`, mas não a `sandhu.pdf` ou `sandhu.txt`?

- A `[sj]*.{.pdf, .txt}`
- B `{s*.pdf, j*.txt}`
- C `[singh,johel]{*.pdf, *.txt}`
- D `{singh.pdf, j*.txt}`

## 4.8 Como posso ordenar as linhas de texto?

Como o próprio nome sugere, o comando `sort`, do inglês *sort* ("ordenar"), coloca os dados em ordem. Por padrão, ele faz isso em ordem alfabética crescente, mas as *flags* `-n` e `-r` podem ser usadas para classificar numericamente e inverter a ordem de sua saída, respectivamente, enquanto `-b` diz para ignorar os espaços em branco à esquerda e `-f` para não diferenciar entre maiúsculas e minúsculas. Os *pipelines* costumam usar o `grep` para se livrar dos registros indesejados e, em seguida, usam o `sort` para colocar os registros restantes em ordem.

## Atividade

Lembra-se da combinação de `cut` e `grep` para selecionar todos os nomes da coluna 1 de `data/csv/biostats.csv`?

```
1 cut -d , -f 1 data/csv/biostats.csv | grep -v Name
```

1. Vá para o diretório inicial.
2. Partindo deste modelo de comando, ordene os nomes da primeira coluna do arquivo `data/csv/biostats.csv` em ordem alfabética decrescente. Para fazer isso, estenda o *pipeline* com uma etapa de ordenação utilizando o `sort`.
3. Agora, utilize o `cut` para selecionar a coluna `Test4` do arquivo `data/csv/grades.csv`, remova o cabeçalho `Test4` utilizando o `grep`, ordene os valores em ordem decrescente (lembre-se de ordenar pelos valores numéricos e não por ordem alfabética) e, por fim, use o `tail` para imprimir as três últimas linhas. Este *pipeline* permite imprimir as três menores notas presentes na coluna `Test4` do arquivo `data/csv/grades.csv`.

## 4.9 Como posso remover linhas duplicadas?

Outro comando frequentemente usado com `sort` é o comando `uniq`, do inglês *unique* (único), cujo trabalho é remover linhas duplicadas. Mais especificamente, ele remove linhas duplicadas adjacentes. Se um arquivo contém:

```
1 2021-07-03
2 2021-07-03
3 2021-08-03
4 2021-08-03
```

então `uniq` produzirá:

```
1 2021-07-03
2 2021-08-03
```

mas se contiver:

```
1 2021-07-03
2 2021-08-03
3 2021-07-03
4 2021-08-03
```

então, o `uniq` imprimirá todas as quatro linhas. A razão é que o `uniq` foi desenvolvido para trabalhar com arquivos muito grandes. Para remover linhas não adjacentes de um arquivo, ele teria que manter todo o arquivo na memória (ou pelo menos todas as linhas únicas vistas até agora). Ao remover apenas duplicatas adjacentes, ele só precisa manter a linha exclusiva mais recente na memória.

## Atividade

Escreva um pipeline para:

1. Vá para o diretório inicial.
2. Obtenha a segunda coluna do arquivo `data/csv/grades.csv`.
3. Remova a palavra "Test2" da saída para que apenas as notas sejam exibidas.
4. Ordene numericamente a saída de modo que todos os valores de notas iguais estejam adjacentes.
5. Exiba cada valor de nota uma única vez (valores apenas uma ocorrência de cada valor) junto com uma contagem de quantas vezes ele ocorre.

O início do *pipeline* é parecido com o do exercício anterior. Estenda-o com um comando `sort` e use `uniq -c` para exibir linhas exclusivas com uma contagem de quantas vezes cada uma ocorre em vez de usar `uniq` e `wc`.

## 4.10 Como posso salvar a saída de um *pipe*?

O Shell nos permite redirecionar a saída de uma sequência de comandos canalizados:

```
1 cut -d , -f 1 data/csv/biostats.csv | grep -v Name > backup/names-only.txt
```

No entanto, `>` deve aparecer no final do *pipeline*: se tentarmos usá-lo no meio, assim:

```
1 cut -d , -f 1 data/csv/biostats.csv > backup/names-only.txt | grep -v Name
```

então, toda a saída de `cut` é gravada em `backup/names-only.txt`, logo não há mais nada para o comando `grep` e ele espera uma eternidade por alguma entrada.

### Pergunta

O que acontecerá se colocarmos o redirecionamento na frente de um *pipeline* como em:

```
1 > result.txt head -n 3 data/csv/grades.csv
```

- A A saída do comando é redirecionada para o arquivo normalmente.
- B O Shell relata isso como um erro.
- C O Shell espera pela entrada para sempre.

## 4.11 Como posso interromper um programa em execução?

Os comandos e *scripts* que você executou até agora foram executados rapidamente, mas algumas tarefas levarão minutos, horas ou até dias para serem concluídas. Você também pode colocar o redirecionamento por engano no meio de um *pipeline*, fazendo com que ele desligue. Se decidir que não deseja que um programa continue em execução, você pode digitar `Ctrl + C` para encerrá-lo. Isso geralmente é escrito `~C` na documentação do Unix; observe que o 'c' pode ser minúsculo.

### Atividade

1. Execute o comando `head` sem argumentos (para que aguarde uma entrada que nunca virá) e, em seguida, pare digitando `Ctrl + C`.

## 4.12 Juntando tudo

Para finalizar, você construirá um *pipeline* para descobrir quantos registros estão no menor dos arquivos de presentes na pasta `data/csv`.



## Atividade

1. Use `wc` com os parâmetros apropriados (*flags*) para listar o número de linhas em todos os arquivos `.csv` da pasta `data/csv`. (Use um caractere curinga para os nomes de arquivos em vez de digitá-los todos manualmente.)
2. Adicione outro comando ao anterior usando um *pipe* para remover a linha que contém a palavra "total".
3. Adicione mais dois estágios ao *pipeline* que usam `sort -n` e `head -n 1` para localizar o arquivo que contém o menor número de linhas.

## Referências

- [1] William Shotts. *The Linux command line: a complete introduction*. No Starch Press, 2019.
- [2] Brian Ward. *How Linux works: What every superuser should know*. no starch press, 2021.