Técnico em Redes
Trabalhando com Texto

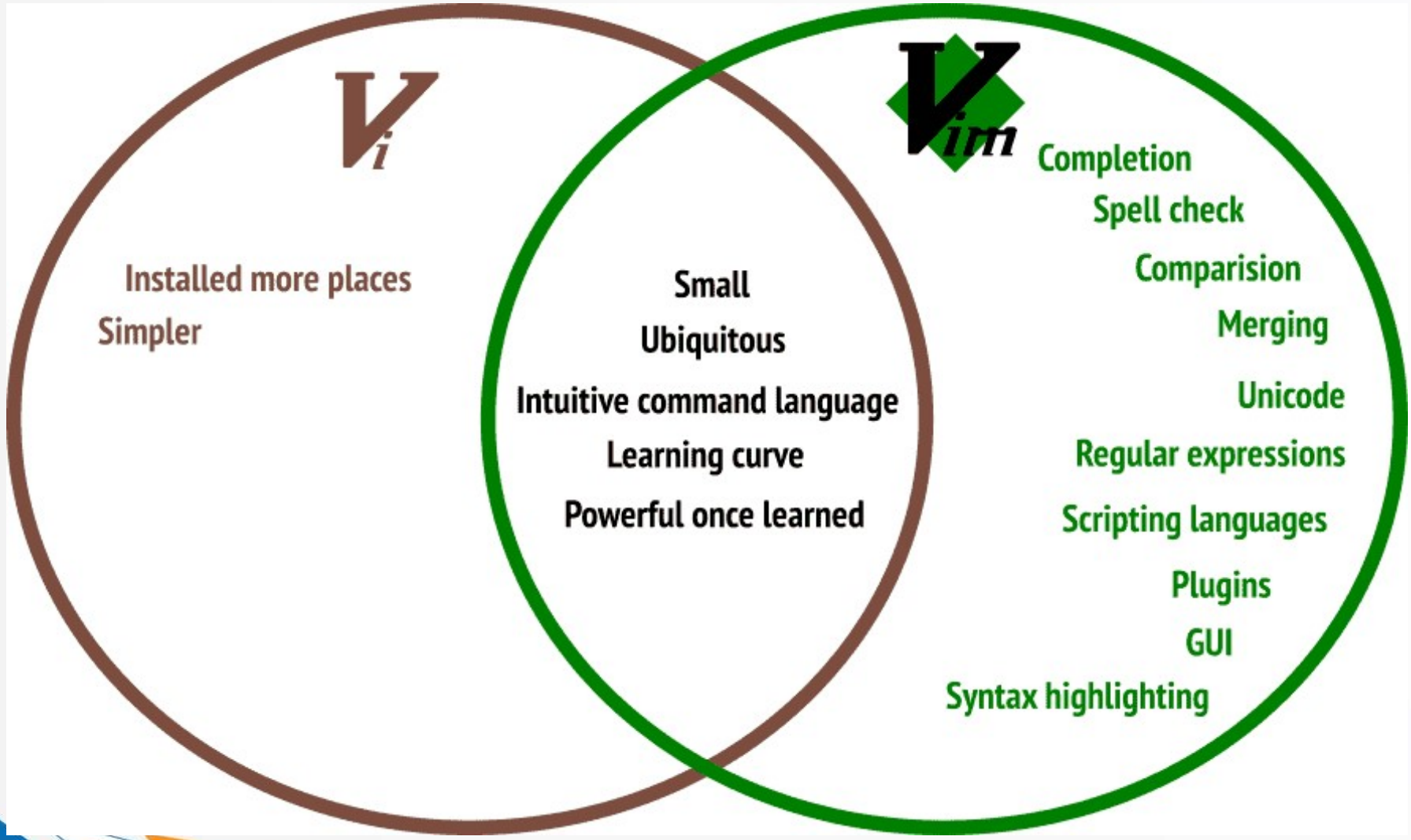
Prof° Marcos Fonseca

Trabalhando com texto

- Um grande número de arquivos em um sistema de arquivos típico são arquivos de texto.
- Os arquivos de texto contêm apenas texto, sem recursos de formatação que você pode ver em um arquivo de processamento de texto.
- Existe um número significativo de comandos para ajudar os usuários a manipular arquivos de texto.
- Também existem recursos disponíveis para o shell controlar a saída dos comandos.
- Em vez de a saída ser exibida na janela do terminal, a saída pode ser redirecionada para outro arquivo ou outro comando.

Editores de Textos no Linux

- Sempre que precisamos digitar textos rápidos ou desenvolver aplicações via linha de comando precisamos de EDITORES DE TEXTO para agilizar nossas tarefas.
- Muitas configurações em sistemas GNU/Linux são feitas editando arquivos de configuração (scripts).
- Existem vários editores de texto para sistemas GNU/Linux. Alguns já vem instados e outros não, mas isso vai depender da distribuição em questão.
- Os Editores mais conhecidos são:
  - **Vi**
  - Vim
  - Nano
  - Gedit
- O Editor que estudaremos a seguir é o "**Vi**"



Diferenças entre Vi e o Vim

Modo edição

- i - insere texto a partir do cursor.
- I - insere texto a partir da primeira palavra da linha atual.
- U - desfaz as alterações realizadas após ter salvo o texto.
- dd - remove (recorta) a linha atual.
- yy - copia a linha atual.
- p - adiciona ao texto informações copiadas ou recortadas.
- : d - remove (recorta) a linha atual.
- D - remove o resto da linha a partir do cursor.
- x - remove o caractere sob o cursor.
- s - remove o caractere sob o cursor e entra em modo de inserção.
- a - entra em modo de inserção na posição seguinte ao cursor.

- : w - salvar
- : w foo - salva o texto num arquivo de nome foo
- : wq - salva e fecha o VI.
- ZZ - salva e sai fecha o VI.
- :q - fecha o VI
- :q! - fecha o VI ignorando qualquer alteração não salva.

Modo Navegação

- e - avança para o fim da palavra.
- w - avança para o inicio da palavra.
- b - retrocede para o inicio da palavra.
- gg - vai para a primeira linha
- G - vai para a última linha.
- HOME - vai para o inicio da linha
- END - vai para o fim da linha
- PAGE DOWN - desce uma tela
- PAGE UP - sobe uma tela

Visualizando arquivos no terminal

- O comando **cat** é um comando útil que pode ser usado para criar e exibir arquivos de texto, bem como combinar cópias de arquivos de texto.
- Para exibir um arquivo digite **cat** e o comando seguido do nome do arquivo:

```
sysadmin@localhost:~/Documents$ cat food.txt
Food is good.
```

- A saída padrão do comando cat é o terminal
- O comando cat também pode ser usado para redirecionar o conteúdo do arquivo para outros arquivos ou inserir outro comando usando caracteres de redirecionamento.

Visualizando arquivos no terminal

- O comando **cat** é bom para visualizar arquivos pequenos, mas não é ideal para arquivos grandes.
- Para visualizar arquivos maiores, use um comando de *pager*.
- Existem dois comandos de *pager* comumente usados:
  - O comando **less** fornece capacidade de paginação avançada, mas não está incluído em todas as distribuições Linux.
  - O comando **more** possui menos recursos que **less**, mas está disponível em distribuições Linux.
- Os comandos more e **less** permitem que os usuários naveguem em um documento usando comandos de teclas.

Comandos de movimento do página

- Para visualizar um arquivo com o comando **less**, passe o nome do arquivo como argumento:

```
sysadmin@localhost:~/Documents$ less words
```

- Ao visualizar um arquivo com o comando less , use a tecla H ou Shift + H para exibir uma tela de ajuda
- Abaixo estão os comandos de movimento mais comumente usados (idênticos em mais e em menos):

Key	Movement
Spacebar	Window forward
B	Window backward
Enter	Line forward
Q	Exit
H	Help

Exibição por segmentos

- Os comandos head e tail são usados para exibir apenas as primeiras ou últimas linhas de um arquivo.
- Passar um número como uma opção fará com que os comandos **head** e **tail** produzam o número especificado de linhas em vez do dez padrão.
- A opção -n pode ser usada para indicar quantas linhas serão geradas.

```
sysadmin@localhost:~/Documents$ head -n5 words
```

Comandos com Pipe

- O *pipe* | caractere pode ser usado para enviar a saída de um comando para outro.
- O caractere de barra vertical permite que você utilize os comandos **head** e **tail** não apenas em arquivos, mas na saída de outros comandos.
- Por exemplo, alguns comandos possuem grande quantidade de saída. Para visualizar mais facilmente o início da saída, canalize-o para o comando head .
- O exemplo a seguir exibe apenas as primeiras dez linhas:

```
sysadmin@localhost:~$ ls /etc | head
```

- Vários pipes podem ser usados consecutivamente para vincular vários comandos. Cada comando vê apenas a entrada do comando anterior.

Redirecionamento Entrada/Saída

- O redirecionamento de entrada/saída (E/S) permite que informações de linha de comando sejam transmitidas para diferentes fluxos.
- Os fluxos são:
  - **STDIN**: Entrada padrão, ou STDIN, são informações inseridas normalmente pelo usuário por meio do teclado.
  - **STDOUT**: A saída padrão, ou STDOUT, é a saída normal dos comandos.
  - **STDERR**: Erro padrão, ou STDERR, são mensagens de erro geradas por comandos.
- O redirecionamento de E/S permite ao usuário redirecionar STDIN para que os dados venham de um arquivo e STDOUT/STDERR para que a saída vá para um arquivo.
- O redirecionamento é obtido usando os caracteres de seta < >.

Redirecionamento de Saída

- A saída padrão de um comando será exibida na tela:

```
sysadmin@localhost:~$ echo "Line 1"
Line 1
```

- Usando o caractere **>** , a saída de um comando pode ser redirecionada para um arquivo:

```
sysadmin@localhost:~$ echo "Line 1" > example.txt
```

- O arquivo example.txt contém a saída do comando echo, que pode ser visualizada com o comando **cat**:

```
sysadmin@localhost:~$ cat example.txt
Line 1
```

Redirecionamento de Saída

- A seta única substitui qualquer conteúdo de um arquivo existente.

- É possível preservar o conteúdo de um arquivo existente usando os caracteres **>>** , que são anexados a um arquivo em vez de substituí-lo.

```
sysadmin@localhost:~$ echo “linha 1" > example.txt
sysadmin@localhost:~$ cat examplo.txt
linha 1
sysadmin@localhost:~$ echo “outra linha” >> example.txt
sysadmin@localhost:~$ cat example.txt
linha 1
Outra linha
```

Direcionamento saída de erro

- STDERR pode ser redirecionado de forma semelhante a STDOUT.
- Usando o caractere > para redirecionar, o fluxo nº 1 é assumido por padrão. Fluxo é a saída do comando.
- O fluxo #2 deve ser especificado ao redirecionar STDERR colocando o número 2 antes do caractere seta >.
- No exemplo, 2> indica que todas as mensagens de erro devem ser enviadas para o arquivo error.txt:

```
sysadmin@localhost:~$ ls /fake
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$ ls /fake 2> erro.txt
sysadmin@localhost:~$ cat erro.txt
ls: cannot access /fake: No such file or directory
```

Redirecionamento múltiplos

- É possível direcionar STDOUT e STDERR de um comando ao mesmo para tanto basta digitar o **e comercial** **&** caractere na frente da seta **>**.
- O conjunto de caracteres &> que significa 1> e 2>:

```
sysadmin@localhost:~$ ls /fake /etc/ppp
ls: cannot access /fake: No such file or directory
/etc/ppp:
ip-down.d  ip-up.d
sysadmin@localhost:~$ ls /fake /etc/ppp &> all.txt
sysadmin@localhost:~$ cat all.txt
ls: cannot access /fake: No such file or directory
/etc/ppp:
ip-down.d
ip-up.d
```

- Se você não quiser que STDERR e STDOUT vão para o mesmo arquivo, eles podem ser redirecionados para arquivos diferentes usando > e 2>.

```
sysadmin@localhost:~$ ls /fake /etc/ppp > examplo.txt 2> erro.txt
```

Redirecionamento de Entrada

- É possível eu redirecionar/inserir/remover dados para dentro de um arquivo sem entrar na edição do mesmo.
- No exemplo abaixo, usando o **tr**, que manda substituir a cadeia de caracteres de A a Z minúsculos por maiúsculo. Para isso invertemos o caractere **<** :

```
sysadmin@localhost:~$ cat example.txt
/etc/ppp:
ip-down.d
ip-up.d
sysadmin@localhost:~$ tr 'a-z' 'A-Z' < example.txt
/ETC/PPP:
IP-DOWN.D
IP-UP.D
```

Classificando arquivos para entrada

- O comando **sort** pode ser usado para reorganizar as linhas de arquivos ou inserir em ordem numérica ou de dicionário.

```
sysadmin@localhost:~$ cat passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
sysadmin@localhost:~$ sort passwd
bin:x:2:2:bin:/bin:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/bin/sh
```

Classificando arquivos para entrada

- O comando **sort** pode reorganizar a saída com base no conteúdo de um ou mais campos.
- Os campos são determinados por um delimitador de campo contido em cada linha.

```
bin:x:2:2:bin:/bbin:x:2:2:bin:/bin:/bin/sh
```

- Por exemplo, o comando a seguir pode ser usado para classificar numericamente o terceiro campo do arquivo passwd.
- A opção **-t** permitirá que outro separador de campo seja especificado.
- A opção **-n** é usada para realizar uma classificação numérica.
- Para especificar por qual campo classificar, use a opção **-k** com um argumento para indicar o número do campo

```
sysadmin@localhost:~$ sort -t: -n -k3 mypasswd
```

Visualizando estáticas dos arquivos

- O comando **wc** mostra o número de linhas, palavras e a contagem de caracteres de um arquivo

```
sysadmin@localhost:~$ wc passwd
  28   37 1455 passwd
```

- É possível visualizar apenas estatísticas específicas:
  - Use a opção -l para mostrar apenas o número de linhas
  - A opção -w para mostrar apenas o número de palavras
  - A opção -c para mostrar apenas o número de bytes ou qualquer combinação dessas opções

Filtrar seções de arquivo

- O comando **cut** pode extrair colunas de texto de um arquivo ou entrada padrão.
- O comando cut é usado para trabalhar com arquivos delimitados – que contêm colunas separadas por um delimitador.
  - A opção **-d** pode especificar delimitadores alternativos, como dois pontos ou vírgula.
  - A opção **-f** pode especificar quais campos serão exibidos.
  - A opção **-c** é usada para extrair colunas de texto com base na posição do caractere.
- No exemplo a seguir, o primeiro, o quinto, o sexto e o sétimo campos do arquivo de banco de dados passwd são exibidos:

```
sysadmin@localhost:~$ cut -d: -f1,5-7 passwd
```

Filtrar conteúdo do arquivo - GREP

- O comando grep pode ser usado para filtrar linhas em um arquivo ou a saída de outro comando que corresponda a um padrão especificado:

```
sysadmin@localhost:~$ grep --color bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
sysadmin:x:1001:1001:System
Administrator,,,,:/home/sysadmin:/bin/bash
```

- O comando grep pode ser usado com diversas opções para filtrar linhas:
  - A opção -d pode especificar delimitadores alternativos, como dois pontos ou vírgula
  - A opção -f pode especificar quais campos serão exibidos.
  - A opção -c é usada para extrair colunas de texto com base na posição do caractere.

Expressões Regulares

- Expressões regulares, também chamadas de regex, são uma coleção de caracteres normais e especiais usados para encontrar padrões simples ou complexos, respectivamente, em arquivos.
- Caracteres normais são caracteres alfanuméricos que correspondem entre si.
- Por exemplo, um caractere a corresponderia a um a
- Caracteres especiais têm significados especiais quando usados em padrões por comandos como o comando **grep**.

Expressões Regulares

- Expressões regulares básicas incluem o seguinte:

Character	Matches
.	Qualquer caractere único.
[ ]	Qualquer caractere ou uma cadeia de caractere
*	Zero ou mais do caractere anterior
^	Se for o primeiro caractere do padrão, o padrão deve estar no início da linha para corresponder, caso contrário, apenas um literal ^
$	Se for o último caractere do padrão, o padrão deve estar no final da linha para corresponder, caso contrário, apenas um literal $

Filtrando a Entrada

- Expressões regulares podem ser usadas com grep para combinar padrões básicos no texto.
- Caracteres âncora:
  - O primeiro caractere âncora ^ é usado para garantir que o padrão apareça no início.

```
sysadmin@localhost:~/Documents$ grep '^root' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

  - O segundo caractere âncora $ é usado para garantir que o padrão apareça no final.

```
sysadmin@localhost:~/Documents$ grep 'r$' alpha-first.txt
B is for Bear
F is for Flower
```

Filtrando a Entrada

- O . caractere corresponderá a qualquer caractere (exceto nova linha)
- Por exemplo, o padrão r..f retornará o seguinte:

```
sysadmin@localhost:~/Documents$ grep 'r..f' red.txt
reef
roof
```

- Os colchetes [ ] correspondem a um único caractere de uma lista ou intervalo contido nos colchetes:

```
sysadmin@localhost:~/Documents$ grep '[0-9]' profile.txt
I am 37 years old.
3121991
I have 2 dogs.
123456789101112
```

Filtrando a Entrada

- O caractere * corresponderá a zero ou mais caracteres que o precedem.

- Por exemplo, o padrão e* corresponderá a zero ou mais da letra, e e resultará no seguinte:

```
sysadmin@localhost:~/Documents$  grep 're*d' red.txt
red
reeed
rd
reed
```