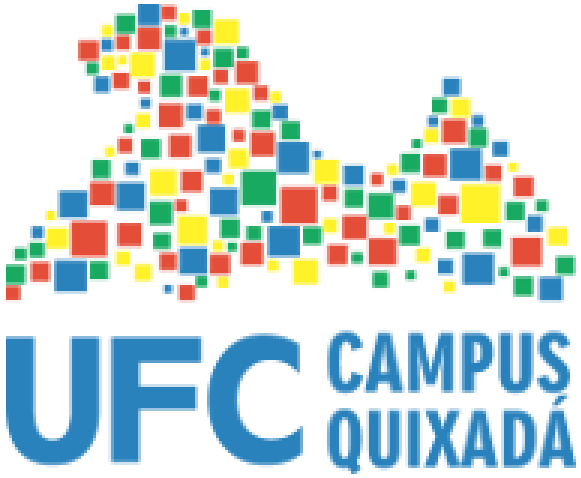

Trabalhando com Textos

Prof. Michel Sales Bonfim

Disciplina: Administração de Sistemas Operacionais Linux

Introdução

- 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.

- Como existem muitos desses arquivos em um sistema Linux típico, existe um número significativo de comandos para ajudar os usuários a manipular arquivos de texto. Existem comandos para visualizar e modificar esses arquivos de várias maneiras.

- Além disso, existem recursos disponíveis para o shell controlar a saída de comandos, portanto, em vez de colocar a saída na janela do terminal, a saída pode ser redirecionada para outro arquivo ou comando. Esses recursos de redirecionamento fornecem aos usuários um ambiente muito mais flexível e poderoso para trabalhar.

Visualizando Arquivos no Terminal

cat

O comando cat, abreviação de concatenate , é um comando simples, mas útil, cujas funções incluem:

1. A criação e exibição de arquivos de texto,
2. Redirecionar o conteúdo de um arquivo para outros arquivos

```
sysadmin@localhost : ~ $ cd Documentos
 sysadmin@localhost : ~/Documentos $ cat food.txt
Comida é boa.
```

Visualizando arquivos com paginação

- Em arquivos grandes, o uso do cat não é recomendado. Para isso, temos os comandos que fazem a leitura com paginação. São eles:
  - O comando **`less`** fornece um recurso de paginação muito avançado.
    - Geralmente é o paginador padrão usado por comandos como o comando **`man`**.
    - Aceita comandos de busca!!!!!
  - O comando **`more`** existe desde os primeiros dias do UNIX.
    - Embora tenha menos recursos do que o comando **`less`**, o comando **`less`** não está incluído em todas as distribuições Linux. O comando **`more`** está sempre disponível.

head & tail

Os comandos `head` e `tail` são usados para exibir apenas as primeiras ou últimas linhas de um arquivo, respectivamente (ou, quando usado com um canal, a saída de um comando anterior). Por padrão, os comandos `head` e `tail` exibem dez linhas do arquivo fornecido como argumento.

```
sysadmin@localhost : ~ $ head -n 3 /etc/sysctl.conf
#
# /etc/sysctl.conf - Arquivo de configuração para definir variáveis do sistema
# Veja /etc/sysctl.d/ para variáveis adicionais do sistema
```

```
sysadmin@localhost : ~ $ tail -5 /etc/sysctl.conf
# Protege contra a criação ou seguimento de links sob certas condições
# Os kernels Debian foram definidos como 1 (restrito)
# Veja https://www.kernel.org/doc/Documentation/sysctl/fs.txt
#fs.protected_hardlinks=0
#fs.protected_symlinks=0
```

Redirecionamentos e Pipes

Introdução

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

Introdução

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

Redirecionamentos e Pipes

Redirecionamentos

>

- Redireciona a saída padrão de um programa/comando/script para algum **dispositivo** ou **arquivo** ao invés do dispositivo de saída padrão (tela).
- Quando é usado com arquivos, este redirecionamento cria ou substitui o conteúdo do arquivo.
- Exemplo:
  - *ls > teste.txt :* Envia a saída do comando ls para o arquivo teste.txt.
  - *ls > /dev/tty2 :* Envia a saída do comando ls para o segundo console.

>>

- Redireciona a saída padrão de um programa/comando/script para algum **dispositivo** ou **adiciona as linhas ao final do arquivo** ao invés do dispositivo de saída padrão (tela).
- Quando é usado com arquivos, este redirecionamento cria ou substitui o conteúdo do arquivo.
- Exemplo:
  - ***ls >> teste.txt*** *:* Adiciona a saída do comando ls ao final do arquivo teste.txt, se ele existir.
  - ***ls >> /dev/tty2*** *:* Envia a saída do comando ls para o segundo console.

<

- Redireciona a entrada padrão de arquivo/dispositivo para um comando. Este comando faz o contrário do anterior, ele envia dados ao comando.

- *Exemplos:*
  - *cat < teste.txt :* enviar o conteúdo do arquivo teste.txt ao comando cat que mostrará seu conteúdo

<

```
sysadmin@localhost:~$ tr 'a-z' 'A-Z'
watch how this works
WATCH HOW THIS WORKS
```

```
sysadmin@localhost:~$ cat example.txt
/etc/ppp:
ip-down.d
ip-up.d
sysadmin@localhost:~$ tr 'a-z' 'A-Z' example.txt
tr: extra operand `example.txt'
Try `tr --help' for more information
```

```
sysadmin@localhost:~$ tr 'a-z' 'A-Z' < example.txt
/ETC/PPP:
IP-DOWN.D
IP-UP.D
```

<<

- Concatena a entrada-padrão
- Este redirecionamento serve principalmente para marcar o fim de exibição de um bloco.
- Este é especialmente usado em conjunto com o comando cat, mas também tem outras aplicações.
- Exemplo:
  - cat << final (este arquivo será mostrado até que a palavra final seja localizada no inicio da linha final

Redirecionamentos e Pipes

Pipes

| (Pipe)

- Envia a saída de um comando para a entrada do próximo comando para continuidade do processamento.
- Os dados enviados são processados pelo próximo comando que mostrará o resultado do processamento.

```
comando1 | comando2
```

Exemplos:

ls -la | more : este comando faz a listagem longa de arquivos que é enviado ao comando more (que tem a função de efetuar uma pausa a cada 25 linhas do arquivo).

locate find | grep "bin/" : neste comando todos os caminhos/arquivos que contém find na listagem serão mostrados (inclusive man pages, bibliotecas, etc.), então enviamos a saída deste comando para grep "bin/" para mostrar somente os diretórios que contém binários.

locate find | grep "bin/" | more

tee

- Envia ao mesmo tempo o resultado do programa para a saída padrão (tela) e para um arquivo.
- Este comando deve ser usado com o pipe “|”.

```
comando |tee [arquivo]
```

Exemplos:

ls -la | tee teste.txt : a saída do comando será mostrada normalmente na tela e ao mesmo tempo gravada no arquivo teste.txt.

Filtrando e Buscando Informações

Ordenação de textos

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

```
sysadmin@localhost:~$ head -5 /etc/passwd > mypasswd
sysadmin@localhost:~$ cat mypasswd
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
```

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

Ordenação de textos

- 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.
  - Na computação, um delimitador é um caractere que separa uma sequência de texto ou dados; o padrão é espaço em branco, como espaços ou tabulações.

`-t[delimitador]`	especifica o *delimitador de campo* . Se o arquivo ou entrada for separado por um delimitador diferente de espaço em branco
`-k3`	especifica o *número do campo* . Por exemplo a opção -k3 faz a classificação pelo terceiro campo.
`-n`	é usada para realizar uma classificação numérica.
`-r`	classificação reversa

Ordenação de textos

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

```
sysadmin@localhost:~$ sort -t: -n -r -k3 mypasswd
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
root:x:0:0:root:/root:/bin/bash
```

Filtrar Seções de Texto

- O comando **`cut`** pode extrair colunas de texto de um arquivo ou entrada padrão. É usado principalmente para trabalhar com arquivos de banco de dados **delimitados**.
- Por padrão, o comando **`cut`** espera que sua entrada seja separada pelo caractere de tabulação, mas a opção **`-d`** pode especificar delimitadores alternativos, como dois pontos ou vírgula.
- A opção **`-f`** pode especificar quais campos exibir, como um intervalo hifenizado ou uma lista separada por vírgulas.

```
sysadmin@localhost : ~ $ cut -d: -f1,5-7 mypasswd
root:root:/root:/bin/bash
daemon:daemon:/usr/sbin:/usr/sbin/nologin
bin:bin:/bin: /usr/sbin/nologin
sys:sys:/dev:/usr/sbin/nologin
sincronização:sync:/bin:/bin/sync
```

Filtrar Seções de Texto

- O comando **`cut`** também é capaz de extrair colunas de texto com base na posição dos caracteres com a opção **`-c`**
  - Útil ao trabalhar com arquivos de banco de dados de largura fixa ou saídas de comando.

```
sysadmin@localhost : ~ $ ls -l | cut -c1-11,50-
total 44
drwxr-xr-x Desktop
drwxr-xr-x Documentos
drwxr-xr-x Downloads
drwxr-xr-x Música
drwxr-xr-x Imagens
drwxr-xr-x Público
drwxr-xr- x Modelos
drwxr-xr-x Vídeos
-rw-rw-r-- all.txt
-rw-rw-r-- example.txt
-rw-rw-r-- mypasswd
-rw-rw-r-- new.txt
```

Filtrar Conteúdo do Texto

- O comando **grep** pode ser usado para filtrar linhas em um arquivo ou na saída de outro comando que corresponda a um padrão especificado.
- Esse padrão pode ser tão simples quanto o texto exato que você deseja corresponder ou pode ser muito mais avançado através do uso de **expressões regulares**.

```
sysadmin@localhost : ~ $ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/ bash
sysadmin:x:1001:1001:Administrador do sistema,,,,:/home/sysadmin:/ bin / bash
```

grep

- Procura por um texto dentro de um arquivo(s) ou no dispositivo de entrada padrão.

```
grep [opções] [expressão] [arquivo]
```

Onde:

expressão palavra ou frase que será procurada no texto. Se tiver mais de 2 palavras você deve identifica-la com aspas “” caso contrário o grep assumirá que a segunda palavra é o arquivo!

arquivo Arquivo onde será feita a procura.

Opções

-A [número] Mostra o [número] de linhas após a linha encontrada pelo grep.

-B [número] Mostra o [número] de linhas antes da linha encontrada pelo grep.

Filtrar Conteúdo do Texto

- A opção -c fornece uma contagem de quantas linhas correspondem

```
sysadmin@localhost:~$ grep -c bash /etc/passwd
2
```

- A opção -n do grep exibirá os números das linhas originais.

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

Filtrar Conteúdo do Texto

- A opção -v inverte a correspondência, exibindo todas as linhas que não contêm o padrão.

```
sysadmin@localhost:~$ grep -v nologin /etc/passwd
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
operator:x:1000:37::/root:/bin/sh
sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin:/bin/bash
```

- A opção -i ignora as distinções de maiúsculas e minúsculas.

```
sysadmin@localhost:~/Documents$ grep -i the newhome.txt
There are three bathrooms.
**Beware** of the ghost in the bedroom.
The kitchen is open for entertaining.
**Caution** the spirits don't like guests.
```

- A opção -w retorna apenas linhas que contêm correspondências que formam palavras inteiras.

Expressões Regulares Básicas no Linux

Expressões regulares fornecem um método poderoso, flexível e eficiente para processar texto.

^	Início de uma string.
$	Fim de uma string.
.	Qualquer caractere (exceto \ n nova linha)
[...]	Conjunto explícito de caracteres para correspondência.
*	0 ou mais da expressão anterior.
[^...]	Nenhum dos caracteres definidos
\	Precedendo um dos itens acima, ele é literal, em vez de um caractere especial. Outros consultar o site

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

Expressões Regulares Estedidas

?	Corresponde ao caractere anterior zero ou uma vez, portanto é um caractere opcional
|	Alternação ou como um operador lógico "ou"
+	Corresponde ao caractere anterior repetido uma ou mais vezes

```
sysadmin@localhost:~/Documents$ grep -E 'colou?r' spelling.txt
American English: Do you consider gray to be a color or a shade?
British English: Do you consider grey to be a colour or a shade?
```

Buscando Arquivos com Expressões Curinga

Caractere	Utilização	Exemplo
*	Coincide com qualquer número de caracteres. Pode ser utilizado como o primeiro ou o último caractere da seqüência de caracteres.	qu* encontra que, quando e quanto
?	Coincide com qualquer caractere alfabético isolado.	B?la localiza bala, bola e bula
[ ]	Coincide com qualquer caractere que esteja entre os colchetes.	B[ao]la localiza bala e bola, mas não bula
!	Coincide qualquer caractere que não esteja entre os colchetes.	b[!ae]la localiza bola e bula, mas não bela

```
sysadmin@localhost:~$ls -d /etc/x*
/etc/xdg
```

find

- Procura por arquivos/diretórios no disco. find pode procurar arquivos através de seu nome, tamanho, tipo, etc através do uso de opções.

```
find [diretório] [opções/expressão]
```

Onde:

diretório Inicia a procura neste diretório, percorrendo seu sub-diretórios.

opções/expressão

-name [expressão] Procura pelo nome [expressão] nos nomes de arquivos e diretórios processados.

-size [num] Procura por arquivos que tiverem o tamanho [num]. [num] pode ser antecedido de “+” ou “-” para especificar um arquivo maior ou menor que [num]. A opção -size pode ser seguida de: k - Especifica o tamanho em Kbytes.

-type [tipo] Procura por arquivos do [tipo] especificado. Os seguintes tipos são aceitos: d – diretório , f - arquivo regular

find

- Exemplo
- ***find / -name teste*** - Procura no diretório raíz e sub-diretórios um arquivo/diretório chamado teste.
- ***find . -size +1000k*** - Procura no diretório atual e sub-diretórios um arquivo com tamanho maior que 1000 kbytes (1Mbyte).

Visualizando estatísticas dos arquivos

- O comando **wc** fornece o número de linhas, palavras e bytes (1 byte = 1 caractere em um arquivo de texto) para um arquivo e uma contagem total de linhas se mais de um arquivo for especificado.
- Por padrão, o comando **wc** permite imprimir até três estatísticas para cada arquivo fornecido, bem como o total dessas estatísticas se for fornecido mais de um nome de arquivo:

```
sysadmin@localhost:~$ wc /etc/passwd /etc/passwd-
  35   56 1710 /etc/passwd
  34   55 1665 /etc/passwd-
  69  111 3375 total
```

df

- Mostra o espaço livre/ocupado de cada partição.

```
df [opções]
```

Onde:

opções

-h, --human-readable Mostra o espaço livre/ocupado em MB, KB, GB ao invés de blocos.

du

- Mostra o espaço ocupado por arquivos e sub-diretórios do diretório atual.

```
du [opções] [diretórios/arquivos]
```

Onde:

diretórios/arquivos elementos para a identificação do tamanho.

opções

-a, --all Mostra o espaço ocupado por todos os arquivos.

-c, --total Faz uma totalização de todo espaço listado.

-h, --human Mostra o espaço ocupado em formato legível por humanos (Kb, Mb) ao invés de usar blocos