Capítulo 3

Sistema de Arquivos e Diretórios

3.1. Objetivos

- Entender o que é FHS;
- Conhecer a estrutura de diretórios do sistema;
- Descobrir alguns diretórios e suas determinadas finalidades.

3.2. Introdução teórica

Quem já teve algum contato com o GNU/Linux, mesmo que superficial, deve ter percebido a presença de vários diretórios (pastas) no sistema. Entretanto, eles estão organizados de uma forma talvez não muito familiar. Neste capítulo, vamos conhecer a organização e explorar a estrutura de diretórios de um sistema GNU/Linux.

Desde que o GNU/Linux foi criado, muito se tem feito para seguir um padrão em relação à estrutura de diretórios. O primeiro esforço para padronização de sistemas de arquivos para o GNU/Linux foi o ***FSSTND - Filesystem Standard***, lançado no ano de 1994.

Cada diretório do sistema tem seus respectivos arquivos que são armazenados conforme regras definidas pela ***FHS - Filesystem Hierarchy Standard***, ou Hierarquia Padrão do Sistema de Arquivos, que define que tipo de arquivo deve ser guardado em cada diretório. Isso é muito importante, pois o padrão ajuda a manter compatibilidade entre as versões Linux existentes no mercado, permitindo que qualquer software escrito para o GNU/Linux seja executado em qualquer distribuição desenvolvida de acordo com os padrões FHS.

Atualmente, o FHS está na sua versão 2.3, e é mantido pelo *Free Standard Group*, uma organização sem fins lucrativos formada por grandes empresas como HP, IBM, Red Hat e Dell.

É vital o compreendimento da FHS para prova, afinal é com ela que nós devemos fazer nossas atividades do dia-a-dia.

3.3. Estrutura de Diretórios GNU/Linux

A estrutura de diretórios também é conhecida como ``Árvore de Diretórios'' porque tem a forma de uma árvore. Mas, antes de estudarmos a estrutura de diretórios, temos que ter em mente o que são diretórios.

Um diretório nada mais é do que o local onde os arquivos são guardados no sistema. O arquivo pode ser um texto, uma imagem, planilha, etc. Os arquivos devem ser identificados por nomes para que sejam localizados por quem deseja utilizá-los.

Um detalhe importante a ser observado é que o GNU/Linux é *case sensitive*, isto é, ele diferencia letras maiúsculas e minúsculas nos arquivos e diretórios.

Sendo assim, um arquivo chamado Arquivo é diferente de **ARQUIVO** e diferente de **arquivo**.

A árvore de diretórios do GNU/Linux tem a seguinte estrutura:

/

bin cdrom etc lib mnt proc root var sys

boot dev home media opt sbin srv tmp usr

Da estrutura mostrada acima, o FHS determina que um sistema GNU/Linux deve conter obrigatoriamente 14 diretórios, especificados a seguir:

/ (raiz)

Este é o principal diretório do GNU/Linux, e é representado por uma ``/'' (barra). É no diretório raiz que ficam todos os demais diretórios do sistema.

Estes diretórios, que vamos conhecer agora, são chamados de subdiretórios pois estão dentro do diretório /.

```
/bin
```

O diretório /bin guarda os comandos essenciais para o funcionamento do sistema.

Esse é um diretório público, sendo assim, os comandos que estão nele podem ser utilizados por qualquer usuário do sistema. Entre os comandos, estão:

- **bash;**
- **ls;**
- **echo;**
- **cp;**

```
/boot
```

No diretório /boot estão os arquivos estáticos necessários à inicialização do sistema, e o gerenciador de boot. .

O gerenciador de boot é um programa que carrega um sistema operacional e/ou permite escolher qual será iniciado.

```
/dev
```

No diretório /dev ficam todos os arquivos de dispositivos. O Linux faz a comunicação com os periféricos por meio de links especiais que ficam armazenados nesse diretório, facilitando assim o acesso aos mesmos.

```
/etc
```

No diretório /etc estão os arquivos de configuração do sistema. Nesse diretório vamos encontrar vários arquivos de configuração, tais como: scripts de

inicialização do sistema, tabela do sistema de arquivos, configuração padrão para logins dos usuários, etc.

```
/lib
```

No diretório /lib estão as bibliotecas compartilhadas e módulos do kernel . As bibliotecas são funções que podem ser utilizadas por vários programas.

```
/media
```

Ponto de montagem para dispositivos removíveis, tais como:

- **cd;**
- **dvd;**
- **disquete;**
- **pendrive;**
- **câmera digital;**

Fique atento: Agora o diretório /media faz parte oficialmente das provas da LPI

```
/mnt
```

Esse diretório é utilizado para montagem temporária de sistemas de arquivos, tais como compartilhamentos de arquivos entre Windows e Linux, Linux e Linux, etc.

```
/opt
```

Normalmente, é utilizado por programas proprietários ou que não fazem parte oficialmente da distribuição.

`/sbin`

O diretório /sbin guarda os comandos utilizados para inicializar, reparar, restaurar e/ou recuperar o sistema. Isso quer dizer que esse diretório também é de comandos essenciais, mas os mesmos são utilizados apenas pelo usuário root.

Entre os comandos estão:

- **halt**
- **ifconfig**
- **init**
- **iptables**

`/srv`

Diretório para dados de serviços fornecidos pelo sistema cuja aplicação é de alcance geral, ou seja, os dados não são específicos de um usuário.

Por exemplo:

- */srv/www (servidor web)*
- */srv/ftp (servidor ftp)*

`/tmp`

Diretório para armazenamento de arquivos temporários. É utilizado principalmente para guardar pequenas informações que precisam estar em algum lugar até que a operação seja completada, como é o caso de um download.

Enquanto não for concluído, o arquivo fica registrado em /tmp, e, assim que é finalizado, é encaminhado para o local correto.

`/usr`

O diretório /usr contém programas que não são essenciais ao sistema e que seguem o padrão GNU/Linux, como, por exemplo, navegadores, gerenciadores de janelas, etc.

O diretório /usr é portável, perceba que dentro dele, existe praticamente uma outra arvore de diretórios independente da primeira, contendo, lib, bin, sbin dentre outras coisas.

```
/var
```

O diretório /var contém arquivos de dados variáveis. Por padrão, os programas que geram um arquivo de registro para consulta, mais conhecido como log, ficam armazenados nesse diretório. Além do log, os arquivos que estão aguardando em filas, também ficam localizados em /var/spool.

Os principais arquivos que se utilizam do diretório /var são :

- mensagens de e-mail;
- arquivos a serem impressos;

3.4. Diretório Recomendado

```
/proc
```

O /proc é um diretório virtual, mantido pelo kernel, onde encontramos a configuração atual do sistema, dados estatísticos, dispositivos já montados, interrupções, endereços e estados das portas físicas, dados sobre as redes, etc.

Aqui, temos subdiretórios com o nome que corresponde ao PID (Process ID) de cada processo.

Dentro deles, vamos encontrar diversos arquivos texto contendo várias informações sobre o respectivo processo em execução.

3.5. O diretório /sys

Pode-se dizer que esse diretório é um primo do diretório /proc. Dentro do diretório /sys podemos encontrar o quase o mesmo conteúdo do proc, mas de uma forma bem mais organizada para nós administradores.

Esse diretório está presente desde a versão 2.6 do kernel e traz novas funcionalidades o que se diz respeito a dispositivos PnP.

3.6. Diretórios Opcionais

Os diretórios /root e /home podem estar disponíveis no sistema, mas não precisam obrigatoriamente possuir este nome.

Por exemplo, o diretório /home poderia se chamar /casa, que não causaria nenhum impacto na estrutura do sistema.

```
/home
```

O /home contém os diretórios pessoais dos usuários cadastrados no sistema.

```
/root
```

Diretório pessoal do super usuário root.

O root é o administrador do sistema, e pode alterar a configuração (dele), configurar interfaces de rede, manipular usuários e grupos, alterar a prioridade dos processos, entre outras.

Dica: Utilize uma conta de usuário normal em vez da conta root para operar seu sistema.

Uma razão para evitar usar privilégios root é por causa da facilidade de se cometer danos irreparáveis como root; além do que, você pode ser enganado e rodar um programa Cavalo de Troia (programa que obtém poderes do super usuário) comprometendo a segurança do seu sistema sem que você saiba.

3.7. Comandos de Movimentação

Vamos aprender agora alguns comandos essenciais para a nossa movimentação dentro do sistema.

O comando pwd exibe o diretório corrente. Ele é muito útil quando estamos navegando pelo sistema e não lembramos qual o diretório atual.

```
# pwd
```

O comando cd é utilizado para mudar o diretório atual de onde o usuário está.

Ir para o diretório home do usuário logado:

```
# cd
# cd ~
```

Ir para o início da árvore de diretórios, ou seja, o diretório / :

```
# cd /
```

Ir para um diretório específico:

```
# cd /etc
```

Sobe um nível na árvore de diretórios:

```
# cd ..
```

Retorna ao diretório anterior:

```
# cd -
# ls
```

Entra em um diretório específico:

```
# cd /usr/include/X11
```

Sobe 2 níveis da árvore de diretórios

```
# cd ../../
```

Atenção! Note a diferença entre caminhos absolutos e relativos:
Absolutos: /etc/ppp; /usr/share/doc; /lib/modules
Relativos: etc/ppp; ../doc; ../../usr;

Fique esperto para conhecer as diferenças entre o **.** *e o* **..** *e o que eles representam para o sistema. Os comandos de movimentação muitas vezes são grandes alvos nas provas, uma boa interpretação desses comandos pode ser necessária, pois você pode precisar deles para resolver uma questão maior.*

3.8. Prática Dirigida

Através dos comandos: cd e pwd, navegue no sistema afim de explorar alguns diretórios.

1) Verificar o diretório atual:

```
$ pwd
```

2) Ir para o início da árvore de diretórios, ou seja, o diretório / :

```
$ cd /
```

3) Ir para o diretório home do usuário logado:

```
$ cd
$ cd ~
```

4) Ir para o diretório /usr/share:

```
$ cd /usr/share
```

5) Subir um nível na árvore de diretórios:

```
$ cd ..
```

6) Retornar ao diretório anterior:

```
$ cd -
```

7) Entre no diretório /var:

```
 $ cd /var
```

8) Entre no diretório /etc e veja o resultado do comando pwd:

```
$ cd /etc
$ pwd
```

9) Utilize o comando cd .., para voltar um nível na hierarquia:

```
 $ cd ..
```

10) Descubra em qual diretório você está através do comando pwd:

```
 $ pwd
```

11) Utilize o comando cd ~, para voltar para seu diretório pessoal:

```
$ cd ~
```

12) Descubra em qual diretório você está através do comando pwd:

```
$ pwd
```

13) Utilize o comando cd -, para voltar ao ultimo diretório acessado:

```
$ cd -
```

14) Descubra em qual diretório você está através do comando pwd:

```
$ pwd
```

3.9. Exercício Teórico

1) Explore os diretórios abaixo, e escreva qual é a função de cada um deles. Justifique:

a) bin

_____

_____

b) boot

_____

_____

c) dev

_____

_____

d) etc

_____

_____

e) home

_____

_____

f) lib

_____

_____

g)media

h) mnt

i) var

j) opt

k) proc

l) root

m) sbin

n) srv

o) tmp

p) usr

2) Qual é a finalidade do comando pwd?

3) Escreva a função de cada um dos comandos abaixo:

a) cd -

b) cd ~

c) cd /

d) cd

e) cd ..

g) cd .

3.10. Laboratório

Alem de todos os diretórios listados acima, na raiz do sistema existe um diretório chamado lost+found, o que representa esse diretório?

Veja também:

FHS - *http://www.pathname.com/fhs/*

Free Standard Group - *http://www.linux-foundation.org/en/Main_Page*

Capítulo 4

Aprendendo comandos do GNU/Linux

4.1. Objetivos

- Listar diretórios;
- Criar e remover arquivos ;
- Criar e remover diretórios;
- Utilizar os caracteres curingas;
- Utilização de outros comando que fazem a diferença em nosso dia-a-dia.

4.2. Introdução teórica

Comandos são instruções passadas ao computador para executar uma determinada tarefa. No mundo *NIX (Linux,Unix), o conceito de comandos é diferente do padrão MS-DOS. Um comando é qualquer arquivo executável, que pode ser ou não criado pelo usuário.

Uma das tantas vantagens do Linux é a variedade de comandos que ele oferece, afinal, para quem conhece comandos, a administração do sistema acaba se tornando um processo mais rápido.

O shell é o responsável pela interação entre o usuário e o sistema operacional, interpretando os comandos.

É no shell que os comandos são executados.

4.2.1. Explorando o sistema

Veremos agora os comandos básicos para navegação no sistema.

O comando ls é utilizado para listar o conteúdo dos diretórios. Se não for especificado nenhum diretório, ele irá mostrar o conteúdo do diretório onde estamos no momento.

Lista o conteúdo do diretório atual:

```
# ls
```

4.3. O comando ls

O comando ls possui muitos parâmetros, veremos aqui as opções mais utilizadas. A primeira dela é o **-l** que lista os arquivos ou diretórios de uma forma bem detalhada (quem criou, data de criação, tamanho, dono e grupo a qual eles pertencem).

```
 # ls -l /
drwxr-xr-x4 root root 1024 2007-01-15 23:17 boot
```

Veja que a saída desse comando é bem detalhada. Falando sobre os campos, para o primeiro caractere temos algumas opções:

d => indica que se trata de um diretório
l => indica que se trata de um link (como se fosse um atalho - também vamos falar sobre ele depois)
- => hífen, indica que se trata de um arquivo
c => indica dispositivo de caractere
b => indica dispositivo de bloco

O campo rwxr-xr-x lista as permissões, enquanto os campos root indicam quem é o usuário e grupo dono desse diretório que, no nosso caso, é o administrador do sistema, o root. O número antes do dono indica o número de hard links, um assunto abordado apenas em cursos mais avançados.

O campo 1024 indica o tamanho do arquivo, e o campo 2007-01-15 23:17

informa a data e hora em que o diretório foi criado. Finalmente, no último campo temos o nome do arquivo ou diretório listado, que, no nosso exemplo, é o boot.

Com relação a diretórios, é importante ressaltar que o tamanho mostrado não corresponde ao espaço ocupado pelo diretório e seus arquivos e subdiretórios. Esse espaço é aquele ocupado pela entrada no sistema de arquivos que corresponde ao diretório.

A opção a lista todos arquivos, inclusive os ocultos:

```
 # ls -a /root
..aptitude.bashrc.profile .rnd.ssh.vmware
.. .bash_history .kde .qt root_161206 .viminfo .Xauthority
```

Veja que, da saída do comando anterior, alguns arquivos são iniciados por . (ponto). Esses arquivos são ocultos.

No Linux, arquivos e diretórios ocultos são iniciados por um . (ponto).

Lista arquivos de forma recursiva, ou seja, lista também os subdiretórios que estão dentro do diretório /:

```
 # ls -R /
```

4.3.1. Curingas

O significado da palavra curinga no dicionário é o seguinte: carta de baralho, que em certos jogos, muda de valor e colocação na seqüência. No sistema GNU/Linux é bem parecida a utilização desse recurso. Os curingas são utilizados para especificar um ou mais arquivos ou diretórios.

Eles podem substituir uma palavra completa ou somente uma letra, seja para listar, copiar, apagar, etc. São usados três tipos de curingas no GNU/Linux:

** - Utilizado para um nome completo ou restante de um arquivo/diretório;*

? - Esse curinga pode substituir uma ou mais letras em determinada posição;

[padrão] - *É utilizado para referência a uma faixa de caracteres de um arquivo/diretório.*

[a-z][0-9] - *Usado para trabalhar com caracteres de a até z seguidos de um caractere de 0 até 9.*

[a,z][1,0] - *Usado para trabalhar com os caracteres a e z seguidos de um caractere 1 ou 0 naquela posição.*

[a-z,1,0] - *Faz referência do intervalo de caracteres de a até z ou 1 ou 0 naquela posição.*

A diferença do método de expansão dos demais, é que a existência do arquivo ou diretório é opcional para resultado final. Isto é útil para a criação de diretórios. Lembrando que os 3 tipos de curingas mais utilizados (``*,?,[ ]'') podem ser usados juntos. Vejamos alguns exemplos:

Supondo que existam 5 arquivos no diretório /home/usuário. Podemos listá-los:

```
# ls
arq1.txt arq2.txt arq3.txt arq4.new arq5.new
```

Vamos listar todos os arquivos do diretório /home/usuário. Podemos usar o coringa ``*'' para visualizar todos os arquivos do diretório:

```
# cd /home/usuário
# ls *
arq1.txt arq2.txt arq3.txt arq4.new arq5.new
```

Para listarmos todos os arquivos do diretório /home/usuário que tenham “new” no nome:

```
 # ls *new*
arq4.new arq5.new
```

No caso, o comando #ls /tmp/teste/* foi citado, mas não tem muito sentido utilizar esse comando, é importante ressaltar que a utilização do * se aplica para um diretório cheio de arquivos, como mostrado no caso dois, utilizado para procurar o arquivo em específico.

4.3.2. Usando curingas no Shell

Listar todos os arquivos que começam com qualquer nome e terminam com .txt:

```
# ls *.txt
```

Listar todos os arquivos que começam com o nome arq, tenham qualquer caractere no lugar do curinga, e terminem com .txt:

```
# ls arq?.txt
```

Para listar todos os arquivos que começam com o nome arq, tenham qualquer caractere entre o número 1-3 no lugar da 4ª letra e terminem com .txt. Neste caso, se obtém uma filtragem mais exata, pois o curinga especifica qualquer caractere naquela posição e [ ] especifica números, letras ou intervalo que serão usados.

```
# ls arq[1-3].txt
```

Para listar somente arq4.new e arq5.new podemos usar os seguintes métodos:

```
# ls *.new
# ls *new*
# ls arq?.new
# ls arq[4,5].*
# ls arq[4,5].new
```

*O parâmetro **-i** do ls, pode ter um grande valor quando o papo são os inodes. Para verificar a informações de inodes por objeto no sistema lembre-se dessa opção.*

Existem muitas outras maneiras de fazer a mesma coisa mas depende muito de cada um que vai utilizar. A criatividade nesse momento conta muito. No exemplo anterior, a última forma resulta na busca mais específica. O que pretendemos é

mostrar como visualizar mais de um arquivo de uma só vez. O uso de curingas é útil para copiar arquivos, apagar, mover, renomear, e nas mais diversas partes do sistema.

4.4. Criação, movimentação, cópia e remoção de arquivos e diretórios

Para criar um arquivo, podemos simplesmente abrir um editor de texto e salvá-lo. Mas existem outras formas.

Uma das formas mais simples é usando o comando touch:

```
# touch arquivo
```

A grande maioria dos comandos básicos devem fazer parte da sua base sólida de conhecimento, é provável que você precise dele para resolver problemas maiores. O touch alem de criar arquivos pode mudar alguns campos da timestamp como hora e tempo.

O comando **mkdir** é utilizado para criar um diretório no sistema. Um diretório é como uma pasta onde você guarda seus arquivos. Exemplo:

Cria o diretório yago:

```
# mkdir yago
```

Cria o diretório 4linux e o subdiretório alunos:

```
 # mkdir -p 4linux/alunos
```

A opção **-p** irá criar o diretório 4linux e o subdiretório alunos, caso não existam.

O comando **rm** é utilizado para apagar arquivos, diretórios e subdiretórios que estejam vazios ou que contenham arquivos.

Exemplos:

Remove o arquivo teste.txt:

```
# rm teste.txt
```

Remove o arquivo yago.txt pedindo confirmação:

```
# rm -i yago.txt
rm: remove arquivo comum `yago.txt'? y
```

A opção **-i** solicita a confirmação para remover o arquivo yago.txt.

Remove o diretório 4linux:

```
# rm -r 4linux
```

A opção **-r** é recursivo, ou seja, irá remover o diretório 4linux e o seu conteúdo.

Observação: Muita atenção ao usar o comando rm! Uma vez que os arquivos e diretórios são removidos não podem mais ser recuperados!

O comando **rmdir** é utilizado para remover diretórios vazios.

Exemplos:

Remove o diretório yago:

```
# rmdir yago
```

Remove o diretório 4linux e o subdiretório alunos:

```
# rmdir -p hackerteen/alunos
```

O comando **cp** serve para fazer cópias de arquivos e diretórios:

```
# cp arquivo-origem arquivo-destino
# cp arquivo-origem caminho/diretório-destino/
# cp -R diretório-origem nome-destino
# cp -R diretório-origem caminho/diretório-destino/
```

O comando mv serve tanto para renomear um arquivo quanto para movê-lo:

```
# mv arquivo caminho/diretório-destino/
# mv arquivo novo-nome
# mv diretório novo-nome
# mv diretório caminho/diretório-destino/
```

*Um opção do comando **cp** muito útil no nosso dia-a-dia é o **-p**, o que faz com que o cp mantenha o timestamp dos arquivos, assim não modificando seus donos nem suas permissões.*

4.5. Prática Dirigida

1) Listar o conteúdo do diretório /:

```
# ls /
```

2) Listar o conteúdo do diretório /root em formato longo:

```
# ls -l /root/
```

3) Listar somente o diretório /boot em formato longo:

```
# ls -ld /boot/
```

4) Listar todos os arquivos do diretório /root, inclusive os ocultos:

```
# ls -a /root
```

5) Listar o conteúdo do diretório /boot de forma recursiva:

```
# ls -R /boot/
```

6) Criar o diretório estudo dentro do diretório /tmp:

```
# mkdir /tmp/estudo
```

7) Criar a seguinte estrutura de diretórios: /backup/2007/fevereiro

```
# mkdir -p /backup/2007/fevereiro
```

8) Remover o diretório /tmp/estudo utilizando o comando rmdir:

```
# rmdir /tmp/estudo
```

9) Crie os arquivos estudo.txt e alunos.txt dentro de /backup/2007/fevereiro.

```
# touch /backup/2007/fevereiro/estudo.txt
# touch /backup/2007/fevereiro/alunos.txt
```

10) Entre no diretório /backup/2007/fevereiro e copie o arquivo estudo.txt para aula.txt:

```
# cd /backup/2007/fevereiro
# cp estudo.txt aula.txt
```

11) Copie o diretório /backup/2007/fevereiro para /backup/2007/janeiro:

```
# cp -R /backup/2007/fevereiro /backup/2007/janeiro
```

12) Remova o arquivo estudo.txt do diretório /backup/2007/fevereiro: # cd /backup/2007/fevereiro

```
# rm estudo.txt
```

13) Renomeie o arquivo alunos.txt do diretório /backup/2007/fevereiro:

```
# cd /backup/2007/fevereiro
# mv alunos.txt teste.txt
```

14) Mova o diretório /backup/2007/fevereiro para /backup/2007/abril:

```
 # mv /backup/2007/fevereiro /backup/2007/abril
```

15) Utilize o comando stat para descobrir algumas informações importantes:

```
#stat /backup
```

4.6. Exercício Teórico

1) Qual é o comando completo a ser executado para criarmos a estrutura de diretórios /stone/blue/gold?

_____
_____
_____
_____

2) Qual a função do comando rmdir?

_____
_____
_____
_____

3) Por que não devemos executar o comando rm com as flags -r e -f?

_____
_____
_____
_____

4) Qual é a opção do comando cp que copiaria arquivos de um diretório de forma recursiva?

5) Qual a função do comando mv?

6) Qual a função do comando ln?

7) Você é um estagiário muito organizado. Ao checar os arquivos do servidor, percebe que alguns arquivos de configuração e arquivos de log estão espalhados no diretório /root. Quais deveriam ser os diretórios corretos para armazenar esse tipo de arquivo?

4.7. Laboratório

1) Liste os arquivos que terminam com a palavra .conf dentro do diretório /etc;
2) Busque no diretório raiz [/] todos os diretórios que terminem com a letra ``n'';

Capítulo 5

Comandos úteis de linha de comando

5.1. Objetivos

- Conhecer alguns comandos importantes para o dia-a-dia.

5.2. Introdução teórica

No mundo GNU/Linux, a maioria das operações são realizadas por meio de comandos escritos. Em geral, permitem um maior controle e flexibilidade de operações, além de poderem ser incluídos em scripts. Neste capítulo iremos aprender sobre alguns comandos básicos.

5.3. Trabalhando com entrada e saída de dados

Esta parte é extremamente importante, pois se trabalha bastante com isso. Por padrão, a entrada do Shell é o teclado, a saída, a tela, e os erros são exibidos na tela também.

Os termos geralmente usados são:

- Entrada de dados, representada por stdin;
- Saída de dados, representada por stdout;
- Saída de erros, representada por stderr;

Mas isso pode ser mudado com o uso de caracteres de redirecionamento, veja abaixo:

Para mudar saída padrão:

- > - Redireciona a saída em um arquivo apagando o conteúdo anterior (se existir); Exemplo:

```
# ls / > tst
# cat tst
# ls /var > tst
# cat tst
```

- >> - Redireciona a saída no final de um arquivo, preservando-o;

Exemplo:

```
 # ls / >> tst
# cat tst
# ls /var >> tst
# cat tst
```

Comandos auxiliares:

- | (pipe, pronuncia-se paipe): Serve para canalizar saída de dado para outro comando;

5.4. Comandos para paginação

Quando os arquivos são maiores do que a altura da tela do computador, eles são mostrados seqüencialmente e até seu final. Isso ocorre numa velocidade que impede que se consiga ler algo. Para esse tipo de arquivo grande, usamos os comandos de paginação. Eles controlam a maneira que os dados de um arquivo são exibidos, seja permitindo uma navegação elementar ou permitindo atingir porções específicas de um arquivo.

5.4.1. Mostrando o conteúdo e/ou concatenando

O comando cat pode ser utilizado para mostrar o conteúdo de um arquivo. Por exemplo, o comando abaixo mostra o conteúdo do arquivo teste.dat.

```
# cat teste.dat
```

Mas o comando cat pode ser utilizado também para concatenação de arquivos. No primeiro exemplo abaixo, é mostrado na tela o conteúdo dos arquivos teste.dat aux.dat. No segundo exemplo, usamos o operador de redirecionamento da saída padrão de modo que a saída do comando cat seja gravada no arquivo tudo.dat.

```
 # cat teste.dat aux.dat
# cat teste.dat aux.dat > tudo.dat
```

De forma análoga, o comando tac também serve para mostrar o conteúdo e concatenar arquivos. Porém, ele mostra o conteúdo de forma reversa, linha a linha. Em outras palavras, ele imprime primeiramente a última linha do arquivo especificado, e finaliza imprimindo a primeira. O exemplo abaixo, mostra o uso do tac para mostrar o conteúdo reverso do arquivo teste.dat.

```
 # tac teste.dat
```

5.4.2. Controlar o fluxo: more e less

Para que a leitura de um arquivo grande na linha de comando seja possível, podemos usar um editor de textos como o vi ou emacs. Contudo, para uma leitura rápida na linha de comando podemos usar os comandos more e less. O comando more permite a leitura contínua de um arquivo. Sempre que a tela é preenchida, o comando more espera por uma ação do usuário para mostrar mais conteúdo.

Pressionando ENTER uma linha a mais é mostrada, pressionando a barra de espaços uma nova página é mostrada. Não é possível retornar (subir) usando o comando more.

```
 # more /var/log/syslog
```

O comando less é mais sofisticado e permite ir e voltar na leitura de um arquivo.

```
# less /var/log/syslog
```

5.4.3. Porções específicas: head e tail

Freqüentemente, queremos ter acesso a porções específicas de um arquivo. Às vezes queremos apenas as linhas iniciais ou as linhas finais. E às vezes queremos um pedaço definido do arquivo. Para essas necessidades, usamos os comandos head para ler porções superiores de um arquivo e tail para ler as porções inferiores. Para ler as 10 primeiras linhas de um arquivo, podemos usar:

```
# head /var/log/syslog
```

Para ler as 10 últimas linhas de um arquivo, podemos usar:

```
# tail /var/log/syslog
```

Os comandos podem ser combinados usando o | (lê-se pipe). Por exemplo, para ler o pedaço entre as linhas 20 e 40 de um arquivo, podemos usar:

```
# head -n 40 /var/log/syslog | tail -n 20
```

O comando acima lê as 40 primeiras linhas do arquivo /var/log/syslog que são passadas para o comando tail que retorna as 20 últimas linhas deste intervalo (as 20 últimas das 40 primeiras = da 20 a 40).

Apesar de parecerem simples, os comandos head e tail fornecem opções de valiosa utilidade. Um grande exemplo é usar a opção ***-f*** *no tail para verificar*

logs em tempo real.

5.4.4. Contagem: wc

Grande parte dos arquivos de configuração e de dados usa uma linha por registro. A contagem destas linhas pode nos fornecer informações muito interessantes.

Por exemplo, a saída abaixo:

```
# wc /etc/passwd
```

Indica que o arquivo contém 32 linhas, 49 blocos (palavras) e 1528 caracteres.

Caso seja necessário apenas o número de linhas, o comando wc pode ser usado com o parâmetro -l, como abaixo:

```
# wc -l /etc/passwd
```

Outros parâmetros possíveis são -w para blocos (palavras) e -c para caracteres.

5.4.5. Classificação: sort

Para diversas ações como eliminação de itens repetidos e rápida visualização de nomes é interessante que possamos classificar um arquivo texto. Na linha de comando, os arquivos textos podem ser classificados usando o comando sort.

A saída do comando abaixo não segue a ordem alfabética:

```
# cat /etc/passwd
```

Podemos mostrar a saída classificada em ordem alfabética, como abaixo:

```
# sort /etc/passwd
```

O comando sort pode ser modificado usando os parâmetros:

- **-f** não considera se as letras estão em caixa alta ou baixa;

- **-n** classificação numérica;
- **-r** classifica na ordem invertida.

5.4.6. Mostrar algo: echo

O comando echo é usado para ecoar algo na tela ou direcionado para um arquivo. Isso é bastante útil para automação. Na linha de comando o echo é útil para inspecionar variáveis de ambiente, que são parâmetros guardados em memória e que definem o ambiente em uso.

Por exemplo, para saber qual a pasta pessoal definida em $HOME do usuário atual:

```
# echo $HOME
```

Para saber qual o idioma definido no console:

```
# echo $LANG
```

Usando o caractere de redirecionamento >, podemos enviar a saída do comando echo para outro destino:

```
# echo $LANG > /tmp/teste
# cat /tmp/teste
```

No exemplo acima, o arquivo teste contém o valor da variável de ambiente $LANG.

5.5. Filtragem

Uma necessidade constante dos administradores é encontrar informações dentro dos arquivos. Para ilustrar, podemos localizar o texto bash no arquivo /etc/passwd:

```
# grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
saito:x:1000:1000:saito,,,:/home/saito:/bin/bash
postgres:x:108:113:PostgreSQL
administrator,,,:/var/lib/postgresql:/bin/bash
jboss:x:1001:1001:JBoss Administrator,,,:/home/jboss:/bin/bash
```

Outra situação possível é procurar pelas entradas que não possuem bash no arquivo passwd. Para isso, usamos o parâmetro -v (inVerter), que inverte a filtragem do grep:

```
# grep -v bash /etc/passwd
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
```

Outros parâmetros do comando grep:

- **-A** [n] Mostra n linhas depois;
- **-B** [n] Mostra n linhas antes;
- **-h** Omite o nome do arquivo nas buscas;
- **-i** Ignora diferença entre maiúsculas e minúsculas;
- **-n** Mostra o número de cada linha encontrada;
- **-v** Inverte a busca, ou seja, encontra apenas as linhas onde o padrão não existir.

O grep pode ser combinado com a saída de outros comandos com o uso do | (pipe). Por exemplo, no comando abaixo, o grep filtra as linhas de endereços IP da saída do comando ifconfig.

```
# ifconfig | grep end.:
```

E, a seguir, o grep é aplicado para filtrar os últimos usuários logados no primeiro terminal (tty1):

```
# last |grep tty1
root tty1 Thu Feb 22 12:19 - 14:21 (02:01)
root tty1 Thu Feb 22 10:50 - down(00:00)
```

Alguns outros tipos de filtros como o grep também podem ser encontrados - ngrep e pgrep.

5.5.1. Filtrar colunas: cut

O comando cut pode ser muito útil para conseguir listagens a partir de arquivos com separadores de colunas definidos.

Por exemplo, para conseguir a primeira coluna do arquivo /etc/passwd, cujo delimitador de colunas é o sinal :, podemos usar o comando:

```
 # cut -f1 -d: /etc/passwd
root
daemon
bin
```

O comando awk é um primo do cut, mas possui mais recursos e opções para expressões regulares. Há situações que o cut não conseguirá resolver o problema, para elas use ***awk.***

5.5.2. Determinando o tipo de arquivo: file

No Linux, extensões de arquivos têm apenas a função de nos auxiliar a nomear os arquivos, a identificá-los e organizá-los facilmente. Não é a extensão que determina o tipo do arquivo, mas sim o seu conteúdo. Por exemplo, se renomearmos um arquivo imagem de 4Linux.jpg para 4Linux.html, ele continuará sendo um arquivo de imagem JPEG.

O comando file determina o tipo do arquivo analisando o seu próprio conteúdo. O exemplo abaixo mostra o uso deste comando:

```
# file arquivo
```

5.6. Administrativos

5.6.1. Espaço em Disco

Aproxima para a unidade de medida mais próxima, mais legível para o ser humano.

```
$ df -h <arquivo/diretório/partição>
```

Mostra em kilobytes.

```
$ df -k <arquivo, diretório ou partição>
```

Mostra em Megabytes.

```
$ df -m <arquivo, diretório ou partição>
```

5.6.2. Definindo tamanho dos objetos

```
$ du -h <arquivo, diretório ou partição>
```

Aproxima para a unidade de medida mais próxima, mais legível para o ser humano.

```
$ du -b <arquivo, diretório ou partição>
```

Mostra em bytes.

```
$ du -k <arquivo, diretório ou partição>
```

Mostra em kilobytes.

```
$ du -m <arquivo, diretório ou partição>
```

Mostra em Megabytes.

```
$ du -l <arquivo, diretório ou partição>
```

Mostra a quantidade de links que arquivo/diretório/partição tem.

```
$ du -s <arquivo, diretório ou partição>
```

Modo silencioso, ou seja, não mostra subdiretórios.

5.6.3. Mostrar o uso de memória RAM: free

O comando free mostra o consumo de memória RAM e os detalhes sobre uso de memória virtual (SWAP):

```
# free
```

Mais detalhes:

```
# free -m
```

A saída do comando será:

```
 total used free shared buffers cached
Mem: 2066856 950944 1115912 038920 342612
-/+ buffers/cache: 569412 1497444
Swap: 570268 0 570268
```

5.6.4. Mostrar e/ou ajustar a data do sistema: date

O comando date pode ser utilizado para mostrar a data e a hora do sistema, e também para ajustá-las. Há várias formas de se utilizar esse comando. A primeira delas é a mais simples:

```
 # date
```

Esse comando mostra a data e a hora atuais do sistema numa formatação padrão. Pode-se utilizar uma string como parâmetro para formatar a saída. O exemplo abaixo mostra o uso de uma string de formatação e o resultado a seguir. Mais informações sobre as opções de formatação podem ser encontradas nas páginas do manual.

```
 # date +"%H:%M, %d-%m-%Y"
12:44, 27-06-2007
```

A outra utilidade do comando date é ajustar a hora do sistema. Obviamente, isso pode ser feito apenas

pelo usuário administrador. A sintaxe é:

```
 # date mmddHHMMYYYY
```

Onde:

- **mm** - número do mês;
- **dd** - dia do mês;
- **HH** – hora;
- **MM** – minutos;

- **YYYY** – ano.;

5.6.5. Mostrar por quanto tempo o computador está ligado: uptime

O comando uptime mostra por quanto tempo o computador está ligado. Além disso, mostra informações sobre o uso do processador:

```
# uptime
```

A saída do comando será:

```
03:20:37 up 16:35, 3 users, load average: 0.16, 0.27, 0.33
```

5.6.6. Mostrar informações sobre o sistema: uname

O comando uname pode ser usado para mostrar informações sobre a versão do kernel em uso e a arquitetura:

```
# uname -a
```

A saída do comando será:

```
Linux professor 2.6.18-3-686 #1 SMP Mon Dec 4 16:41:14 UTC 2006 i686
GNU/Linux
```

5.6.7. Diferença entre arquivos: diff

O programa diff nos permite verificar a diferença entre arquivos e diretórios. No caso de diretórios, é importante o uso da opção -r para assegurar a comparação de todos os subdiretórios.

```
# diff arquivo1 arquivo2
# diff -r dir1 dir2
```

5.6.8. Tempo de execução de um programa: time

O comando time permite medir o tempo de execução de um programa. Sua sintaxe é:

```
# time programa
```

5.6.9. Localização no sistema: find

O comando find procura por arquivos/diretórios no disco. Ele pode procurar arquivos pela sua data de modificação, tamanho, etc, com o uso de opções. Find, ao contrário de outros programas, usa opções longas por meio de um ``-''.

Sintaxe:

find [diretório] [opções/expressão]

- **-name [expressão] :**

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

```
# find /etc -name *.conf
```

- **-maxdepth [num] :**

Limite a profundidade de busca na árvore de diretórios. Por exemplo, limitando a 1, irá procurar apenas no diretório especificado e não irá incluir nenhum subdiretório.

```
# find /etc -maxdepth 1 -name *.conf
```

- **-amin [num] :**

Procura por arquivos que foram acessados [num] minutos atrás. Caso seja antecedido por ``-'', procura por arquivos que foram acessados entre [num] minutos atrás e o momento atual.

```
# find ~ -amin -5
```

- **-atime [num] :**

Procura por arquivos que foram acessados [num] dias atrás. Caso seja antecedido por ``-``, procura por arquivos que foram acessados entre [num] dias atrás e a data atual.

```
# find ~ -atime -10
```

- **-uid [num] :**

Procura por arquivos que possuem a identificação numérica do usuário igual a [num].

```
# find / -uid 1000
```

- **-user [nome] :**

Procura por arquivos que possuem a identificação de nome do usuário igual a [nome].

```
# find / -user aluno
```

- **-perm [modo] :**

Procura por arquivos que possuem os modos de permissão [modo]. Os [modo] de permissão podem ser numérico (octal) ou literal.

```
# find / -perm 644
```

- **-size [num] :**

Procura por arquivos que tenham o tamanho [num]. O tamanho é especificado em bytes. Você pode usar os sufixos k, M ou G para representar em quilobytes, Megabytes ou Gigabytes. [num] Pode ser antecedido de ``+'' ou ``-'' para especificar um arquivo maior ou menor que [num].

```
# find / -size +1M
```

- **-type [tipo] :**

Procura por arquivos do [tipo] especificado. Os seguintes tipos são aceitos:

- **b -** bloco
- **c** - caractere
- **d -** diretório
- **p** - pipe
- **f -** arquivo regular
- **l-** link simbólico
- **s -** socket

find /dev -type b

Outros exemplos:

```
# find / -name grep
```

Procura no diretório raiz e nos subdiretórios um arquivo/diretório chamado grep.

```
# find / -name grep -maxdepth 3
```

Procura no diretório raiz e nos subdiretórios até o 3º nível, um arquivo/diretório chamado grep.

```
# find . -size +1000k
```

Procura no diretório atual e nos subdiretórios um arquivo com tamanho maior que 1000 kbytes (1Mbyte).

```
# find / -mmin -10
```

Procura no diretório raiz e nos subdiretórios um arquivo que foi modificado há 10 minutos atrás ou menos.

5.6.10. Localização usando base de dados: locate

O comando locate é um comando rápido de busca de arquivos, porém não usa busca recursiva na sua árvore de diretórios. Ele cria uma base de dados para que a busca seja mais rápida pelo comando updatedb, que inclusive poderá ser agendado para que a base de dados esteja sempre atualizada.

Para utilizá-lo, primeiro é necessário criar a sua base de dados usando a seguinte sintaxe:

```
# updatedb
```

Quando esse comando é executado pela primeira vez costuma demorar um pouco. Para o comando locate, usamos a seguinte sintaxe:

```
# locate howto
```

A saída do comando será:

```
/root/Documentação/securing-debian-howto.en.pdf
/usr/share/doc/diveintopython/html/appendix/fdl_howto.html
/usr/share/doc/cacti/html/graph_howto.html
/usr/share/doc/pcmciautils/mini-howto.txt.gz
/usr/share/doc/python2.4-xml/examples/test/output/test_howto
/usr/share/doc/python2.4-xml/examples/test/test_howto.py.gz
/usr/share/doc/python2.4-xml/howto.cls
/usr/share/doc/python2.4-xml/xml-howto.tex.gz
/usr/share/doc/python2.4-xml/xml-howto.txt.gz
/usr/share/vim/vim64/doc/howto.txt
```

5.7. Mais e mais comandos

- **awk** - linguagem de procura de padrões e processamento;
- **logger** comando de interface entre a shell e o syslog;
- **sed** editor de texto linha a linha;
- **seq** - imprime uma sequencia de números;
- **sleep** - insere uma pausa pelo número de segundos especificado;
- **expand** - converte tabs em espaços;
- **unexpand** - converte espaços em tabs;
- **join -** junta a saída de uma arquivo em outro arquivo, jogando na tela;
- **nl** - numera as linhas na saída do comando;
- **paste** - junta os arquivos na saída padrão;
- **split** - usado para dividir determinado arquivo em pedaços menores;
- **tr** - substitui ou remove os caracteres selecionados da entrada padrão para a saída padrão;
- **uniq** - remove linhas desnecessárias ou duplicadas, ou seja, ele faz uma espécie de listagem de cada linha única do arquivo;
- **fmt** - formatador de texto, muito prático quando precisa-se fazer uma formatação rápida em algum arquivo;
- **hexdump** - converte arquivo para hexadecimal, decimal, ASCII;
- **od** - converte arquivo para octal;
- **xargs** - poderoso, e muito bom para listagem de arquivos.

Dica LPI: Atenção:
A maioria desses comandos são bastante simples de ser utilizados e possuem poucos parâmetros, entretanto alguns como awk, sed, find, grep, xargs, uniq, join, paste e cut possuem manuais bastante extensos dada a quantidade de tarefas que podem realizar. Dessa forma a leitura de suas páginas de man são altamente sugeridas.

Os comandos e estrutura de shell script aqui citados são utilizados também no Red Hat.

5.8. Prática Dirigida

Inicialmente vamos executar exemplos de funcionamento dos comandos mais utilizados em Shell Scripts.

1) **Copie o arquivo passwd do sistema para o diretório /tmp:**

```
# cp /etc/passwd /tmp
```

2) **Entre no diretório /tmp para iniciarmos os testes:**

```
# cd /tmp
```

3) **Agora, dentro do diretório /tmp, utilizaremos o awk para separar o conteúdo do arquivo passwd utilizando o caracter ``:'' como separador e imprimir na tela o usuário e sua respectiva shell adicionando a palavra ``uses'' entre uma coluna e outra:**

```
# awk -F : '{print $1 " uses " $7}' passwd
```

4) Utilizemos o comando cat para ver o conteúdo do arquivo passwd:

```
# cat passwd
```

5) Com o comando cut vamos realizar um procedimento análogo ao do awk; entretanto com este comando não é possível inserir a palavra ``uses'' entre as colunas:

```
 # cut -d : -f 1,7 passwd
```

6) Utilizemos o comando date para listar a hora atual do sistema. Em seguida vamos alterá-la e depois modificar o seu padrão de saída:

```
# date
# date mmddHHMMYYYY
# date +%Y%m%d-%H%M
```

7) Determine quanto cada filesystem está utilizando de seu espaço. Depois imprima em formato “humam readable”, ou seja, que um humano entende fácilmente, utilizando o parâmetro -h. Na sequencia verifique qual a porcentagem de inodes utilizados:

```
 # df
# df -h
# df -i
```

8) A única forma de determinar quanto espaço em disco um diretório está ocupando é somando o tamanho de todos os arquivos e arquivos em seus subdiretórios. O comando du realiza essa tarefa:

```
 # du -h
```

9) Para imprimir uma mensagem na tela, basta utilizar o comado echo. Os parâmetros -ne fazem com que o echo interprete caracteres de controle como ;

```
 # echo -ne "Um tab\tseuguido de quebra de linha\n"
```

10) Juntando dois arquivos em um:

```
 # echo "1:Debian Sarge:3.1:Stable" > sarge
# echo "1:Debian Etch:4.0:Stable" > etch
# join -t: sarge etch
```

O comando join (unir) concatena registros de dois arquivos de texto baseado em índices comuns entre os registros. No nosso caso o separador de campos será o caractere dois-pontos (-t:).

11) Comando paste, junta os arquivos na saída padrão. Diferente do join, ele joga os dois arquivos lado-a-lado.:

```
# paste sarge etch
```

Ainda com o paste podemos, usar o parâmetro -d, de delimitador:

```
# paste -d@ sarge etch
```

E também, um em baixo do outro:

```
# paste -s sarge etch
```

12) Vamos procurar por arquivos que satisfaçam o padrão pas* dentro do diretório /etc:

```
# find /etc -name 'pas*'
```

13) Realize a procura dentro do arquivo passwd por todas as linhas que iniciam pelo caracter ``s'':

```
# grep ^s passwd
```

14) Mostre na tela apenas as primeira 15 linhas do arquivo passwd:

```
# head -n 15 passwd
```

15) O comando split é usado para dividir um arquivo em pedaços menores, muito útil quando se tem dois disquetes e um arquivo de 2 Mb:

```
# cp /var/log/messages .
# split --lines=50 messages
```

Isso irá gerar X arquivos com 50 linhas cada:

```
# wc -l x*
```

Apague todos os arquivos:

```
# rm -r x*
```

16) Vamos verificar que a linha foi escrita mostrando as últimas 15 linhas do respectivo arquivo de log:

```
# tail -n 15 /var/log/messages
# tail -n 15 /var/log/syslog
```

Mostre na tela como ficaria o arquivo passwd se substituíssemos todos os caracteres ``:'' pelo padrão ``_MU_'':

```
# sed 's/:/_MU_/g' passwd
```

17) Utilize o comando seq para imprimir a sequencia de números de 1 a 100 pulando de dois em dois:

```
# seq 1 2 100
```

18) Faça a shell aguardar por dez segundos...

```
# sleep 10
```

19) Ordene as linhas do arquivo passwd:

```
# sort passwd
```

20) Vamos substituir e remover os caracteres do arquivo:

```
# tr a-z A-Z < /etc/passwd
# tr -d 0 < /etc/passwd
```

21) Quantas linhas há no arquivo passwd?

```
 # wc -l passwd
```

22) Verifique os usuários que estão logados no sistema:

```
 # who
```

23) Vamos agora listar diretórios utilizando o xargs:

```
# ls / | xargs -n1
# ls / | xargs -n2
# ls / | xargs -n3
```

Você percebeu que no primeiro comando ele listou o diretório, jogando na tela um nome de cada vez. O segundo comando fara o mesmo só que com dois nomes na mesma linha, e o terceiro também.

Outros testes com o xargs:

```
# ls / > teste_xargs.txt
# cat teste_xargs.txt
# cat teste_xargs.txt | xargs -n 2
# xargs -n 3 < teste_xargs.txt
```

24) Comando nl, numera as linhas na saída do comando:

```
# nl /etc/passwd
# grep sys /etc/passwd | nl
# ls -l /etc | nl
# ls -l /etc | tail | nl
```

5.9. Exercícios Teóricos

1) Qual é a principal diferença entre os comandos find e locate?

2) Qual é a função do comando tac?

3) Qual é a função do comando echo?

4) Qual é a diferença entre o comando du e o comando df?

5) Qual é a utilidade do comando time?

5.10. Laboratório

1) Obtenha uma lista de usuários a partir do arquivo /etc/passwd, contendo as colunas referentes ao usuário e seu diretório pessoal;
2) Obtenha uma lista classificada por tamanho dos arquivos da pasta /var/log.