COMANDOS BÁSICOS E INTERMEDIÁRIOS PARA LINUX

ALAN BRAGA

Guia Completo do Usuaíro

Sumário

1. Introdução ao Linux

O Linux é um sistema operacional de código aberto baseado em Unix, amplamente utilizado em servidores, desktops e dispositivos embarcados. Dominar os comandos Linux é essencial para profissionais de TI, desenvolvedores e administradores de sistemas.

1.1 O Terminal

O terminal (ou shell) é a interface de linha de comando onde você interage com o sistema operacional através de comandos textuais. O shell mais comum é o Bash (Bourne Again Shell).

> ■ ***Dica:*** *Pressione Tab para autocompletar comandos e nomes de arquivos. Use as setas ↑ e ↓ para navegar pelo histórico de comandos.*

2. Navegação e Manipulação de Arquivos

2.1 Comandos de Navegação

Comando	Descrição	Exemplo
`pwd`	Mostra o diretório atual	`pwd`
`ls`	Lista arquivos e diretórios	`ls -la`
`cd`	Muda de diretório	`cd /home/usuario`
`cd ..`	Volta um diretório	`cd ..`
`cd ~`	Vai para o diretório home	`cd ~`

2.2 Listagem de Arquivos

Opções úteis do comando **ls**:

```
ls -l
```

Lista em formato longo com permissões, proprietário, tamanho e data.

```
ls -a
```

Mostra arquivos ocultos (que começam com ponto).

```
ls -lh
```

Lista com tamanhos em formato legível (K, M, G).

```
ls -R
```

Lista recursivamente todos os subdiretórios.

2.3 Manipulação de Arquivos e Diretórios

Comando	Descrição	Exemplo
`mkdir`	Cria diretório	`mkdir pasta`
`mkdir -p`	Cria diretórios recursivamente	`mkdir -p dir1/dir2/dir3`
`touch`	Cria arquivo vazio	`touch arquivo.txt`
`cp`	Copia arquivos	`cp origem destino`
`cp -r`	Copia diretórios	`cp -r pasta1 pasta2`
`mv`	Move/renomeia arquivos	`mv antigo.txt novo.txt`
`rm`	Remove arquivos	`rm arquivo.txt`

`rm -r`	Remove diretórios	`rm -r pasta`
`rm -rf`	Remove forçadamente	`rm -rf pasta`
`rmdir`	Remove diretório vazio	`rmdir pasta`

■■ ***Atenção:*** *O comando* `rm -rf` *é extremamente perigoso! Ele remove arquivos sem confirmação e não há lixeira para recuperá-los. Use com muito cuidado!*

3. Gerenciamento de Permissões

No Linux, cada arquivo e diretório possui permissões que determinam quem pode ler, escrever ou executar. As permissões são divididas em três grupos: proprietário (user), grupo (group) e outros (others).

3.1 Entendendo as Permissões

Ao executar `ls -l`, você verá algo como:

```
-rwxr-xr-- 1 usuario grupo 4096 Nov 15 10:30 arquivo.txt
```

Símbolo	Significado
r	Read (leitura) = 4
w	Write (escrita) = 2
x	Execute (execução) = 1
-	Sem permissão

3.2 Modificando Permissões

Comando	Descrição	Exemplo
`chmod`	Altera permissões	`chmod 755 script.sh`
`chmod u+x`	Adiciona execução ao usuário	`chmod u+x arquivo`
`chmod g-w`	Remove escrita do grupo	`chmod g-w arquivo`
`chmod a+r`	Adiciona leitura a todos	`chmod a+r arquivo`
`chown`	Muda proprietário	`chown usuario arquivo`
`chgrp`	Muda grupo	`chgrp grupo arquivo`

3.3 Exemplos Práticos de Permissões

Permissão	Numérico	Significado
`rwxrwxrwx`	`777`	Todos podem tudo (perigoso!)
`rwxr-xr-x`	`755`	Dono pode tudo, outros só leem/executam
`rw-r--r--`	`644`	Dono lê/escreve, outros só leem
`rwx------`	`700`	Apenas o dono tem acesso total
`rw-------`	`600`	Apenas o dono lê/escreve

4. Processos e Sistema

Um processo é um programa em execução. O Linux permite visualizar, gerenciar e controlar processos de forma eficiente.

4.1 Visualização de Processos

Comando	Descrição	Exemplo
`ps`	Lista processos atuais	`ps aux`
`top`	Monitor de processos em tempo real	`top`
`htop`	Monitor interativo (mais amigável)	`htop`
`pgrep`	Busca processos por nome	`pgrep firefox`
`pidof`	Encontra PID de um processo	`pidof chrome`

4.2 Gerenciamento de Processos

Comando	Descrição	Exemplo
`kill`	Encerra processo por PID	`kill 1234`
`kill -9`	Força encerramento	`kill -9 1234`
`killall`	Encerra por nome	`killall firefox`
`pkill`	Encerra por padrão	`pkill -f python`
`&`	Executa em background	`comando &`
`fg`	Traz processo para foreground	`fg`
`bg`	Continua processo em background	`bg`
`jobs`	Lista processos do shell atual	`jobs`
`Ctrl+Z`	Suspende processo atual	-
`Ctrl+C`	Interrompe processo atual	-

4.3 Informações do Sistema

Comando	Descrição	Exemplo
`uname -a`	Informações do kernel	`uname -a`
`hostname`	Nome do host	`hostname`

`uptime`	Tempo ligado e carga	`uptime`
`free -h`	Memória disponível	`free -h`
`df -h`	Espaço em disco	`df -h`
`du -sh`	Tamanho de diretório	`du -sh /home`
`lscpu`	Informações da CPU	`lscpu`
`lsblk`	Lista dispositivos de bloco	`lsblk`

5. Redirecionamento e Pipes

O redirecionamento permite controlar a entrada e saída de comandos, enquanto pipes conectam a saída de um comando à entrada de outro.

5.1 Operadores de Redirecionamento

Operador	Descrição	Exemplo
>	Redireciona saída (sobrescreve)	`ls > lista.txt`
>>	Redireciona saída (anexa)	`echo 'texto' >> arquivo.txt`
<	Redireciona entrada	`sort < lista.txt`
2>	Redireciona erros	`comando 2> erros.txt`
&>	Redireciona saída e erros	`comando &> tudo.txt`
|	Pipe (conecta comandos)	`ls | grep '.txt'`
tee	Escreve em arquivo e tela	`ls | tee lista.txt`

5.2 Exemplos Práticos

Salvar listagem de arquivos:

```
ls -la > listagem.txt
```

Adicionar texto ao final de um arquivo:

```
echo 'Nova linha' >> arquivo.txt
```

Contar linhas de um arquivo:

```
cat arquivo.txt | wc -l
```

Buscar processo e terminar:

```
ps aux | grep firefox | awk '{print $2}' | xargs kill
```

Salvar saída e exibir na tela:

```
ls -l | tee listagem.txt
```

6. Busca e Filtros

6.1 Comando grep

O `grep` é usado para buscar padrões em arquivos de texto.

Comando	Descrição	Exemplo
`grep 'padrão' arquivo`	Busca padrão em arquivo	`grep 'error' log.txt`
`grep -i`	Ignora maiúsculas/minúsculas	`grep -i 'error' log.txt`
`grep -r`	Busca recursivamente	`grep -r 'TODO' .`
`grep -n`	Mostra número da linha	`grep -n 'erro' arquivo.txt`
`grep -v`	Inverte busca (não contém)	`grep -v 'sucesso' log.txt`
`grep -c`	Conta ocorrências	`grep -c 'error' log.txt`
`grep -A 3`	Mostra 3 linhas após	`grep -A 3 'error' log.txt`
`grep -B 3`	Mostra 3 linhas antes	`grep -B 3 'error' log.txt`

6.2 Comando find

O `find` localiza arquivos e diretórios no sistema.

Comando	Descrição	Exemplo
`find . -name`	Busca por nome	`find . -name '*.txt'`
`find . -type f`	Busca apenas arquivos	`find . -type f`
`find . -type d`	Busca apenas diretórios	`find . -type d`
`find . -size +100M`	Arquivos maiores que 100MB	`find . -size +100M`
`find . -mtime -7`	Modificados nos últimos 7 dias	`find . -mtime -7`
`find . -user`	Busca por proprietário	`find . -user joao`
`find . -exec`	Executa comando nos resultados	`find . -name '*.log' -exec rm {} \;`

6.3 Outros Comandos de Filtro

Comando	Descrição	Exemplo
`wc`	Conta linhas, palavras, bytes	`wc -l arquivo.txt`
`sort`	Ordena linhas	`sort arquivo.txt`

`uniq`	Remove linhas duplicadas	`sort arquivo.txt | uniq`
`head`	Mostra primeiras linhas	`head -n 10 arquivo.txt`
`tail`	Mostra últimas linhas	`tail -n 20 log.txt`
`tail -f`	Monitora arquivo em tempo real	`tail -f /var/log/syslog`
`cut`	Extrai colunas	`cut -d: -f1 /etc/passwd`
`awk`	Processa texto por colunas	`awk '{print $1}' arquivo.txt`
`sed`	Editor de stream	`sed 's/antigo/novo/g' arquivo.txt`

7. Compressão e Arquivamento

O Linux oferece diversas ferramentas para comprimir e arquivar arquivos, economizando espaço e facilitando transferências.

7.1 Comando tar

O `tar` (Tape Archive) é usado para criar e extrair arquivos.

Comando	Descrição	Exemplo
`tar -cvf`	Cria arquivo tar	`tar -cvf arquivo.tar pasta/`
`tar -xvf`	Extrai arquivo tar	`tar -xvf arquivo.tar`
`tar -tvf`	Lista conteúdo	`tar -tvf arquivo.tar`
`tar -czvf`	Cria tar.gz	`tar -czvf arquivo.tar.gz pasta/`
`tar -xzvf`	Extrai tar.gz	`tar -xzvf arquivo.tar.gz`
`tar -cjvf`	Cria tar.bz2	`tar -cjvf arquivo.tar.bz2 pasta/`
`tar -xjvf`	Extrai tar.bz2	`tar -xjvf arquivo.tar.bz2`

■ ***Dica:*** *As opções do tar seguem um padrão:* `c` *(create),* `x` *(extract),* `v` *(verbose),* `f` *(file),* `z` *(gzip),* `j` *(bzip2).*

7.2 Outros Comandos de Compressão

Comando	Descrição	Exemplo
`gzip`	Comprime arquivo	`gzip arquivo.txt`
`gunzip`	Descomprime gzip	`gunzip arquivo.txt.gz`
`bzip2`	Comprime arquivo (mais eficiente)	`bzip2 arquivo.txt`
`bunzip2`	Descomprime bzip2	`bunzip2 arquivo.txt.bz2`
`zip`	Cria arquivo zip	`zip arquivo.zip pasta/*`
`unzip`	Extrai arquivo zip	`unzip arquivo.zip`
`7z a`	Cria arquivo 7z	`7z a arquivo.7z pasta/`
`7z x`	Extrai arquivo 7z	`7z x arquivo.7z`

8. Rede e Conectividade

8.1 Comandos de Rede Básicos

Comando	Descrição	Exemplo
`ping`	Testa conectividade	`ping google.com`
`ifconfig`	Configura interface de rede	`ifconfig`
`ip addr`	Mostra endereços IP	`ip addr show`
`ip route`	Mostra rotas	`ip route show`
`netstat`	Estatísticas de rede	`netstat -tuln`
`ss`	Socket statistics (mais moderno)	`ss -tuln`
`hostname`	Mostra/altera nome do host	`hostname`
`host`	Consulta DNS	`host google.com`
`nslookup`	Consulta DNS	`nslookup google.com`
`dig`	Consulta DNS (detalhado)	`dig google.com`

8.2 Transferência de Arquivos

Comando	Descrição	Exemplo
`scp`	Copia arquivos via SSH	`scp arquivo.txt user@host:/path`
`rsync`	Sincroniza arquivos	`rsync -avz origem/ destino/`
`wget`	Baixa arquivos da web	`wget http://site.com/arquivo.zip`
`curl`	Transfere dados de/para servidor	`curl -O http://site.com/arquivo`
`ftp`	Cliente FTP	`ftp servidor.com`
`sftp`	FTP seguro via SSH	`sftp user@host`

8.3 SSH e Conexões Remotas

Comando	Descrição	Exemplo
`ssh`	Conecta remotamente	`ssh user@servidor.com`
`ssh -p`	Conecta em porta específica	`ssh -p 2222 user@servidor`
`ssh-keygen`	Gera par de chaves SSH	`ssh-keygen -t rsa -b 4096`

`ssh-copy-id`	Copia chave pública	`ssh-copy-id user@servidor`
`sshfs`	Monta sistema de arquivos remoto	`sshfs user@host:/path /mnt`

9. Editores de Texto

9.1 Editor nano

O `nano` é um editor de texto simples e amigável, ideal para iniciantes.

Comando	Descrição
`nano arquivo.txt`	Abre/cria arquivo
`Ctrl+O`	Salva arquivo
`Ctrl+X`	Sai do editor
`Ctrl+K`	Recorta linha
`Ctrl+U`	Cola linha
`Ctrl+W`	Busca texto
`Ctrl+\`	Substitui texto
`Ctrl+G`	Ajuda

9.2 Editor vim

O `vim` é um editor poderoso e versátil, com curva de aprendizado maior.

Comando	Descrição
`vim arquivo.txt`	Abre/cria arquivo
`i`	Entra no modo de inserção
`Esc`	Volta ao modo normal
`:w`	Salva arquivo
`:q`	Sai do editor
`:wq ou :x`	Salva e sai
`:q!`	Sai sem salvar
`dd`	Deleta linha
`yy`	Copia linha
`p`	Cola linha
`/texto`	Busca texto
`:%s/antigo/novo/g`	Substitui texto globalmente

■ ***Dica:*** *Se você se perder no vim, pressione* `Esc` *várias vezes, depois digite* `:q!` *e pressione Enter para sair.*

10. Shell Script Básico

Shell script é uma forma de automatizar tarefas através de scripts. É uma habilidade essencial para administradores de sistemas.

10.1 Estrutura Básica

Todo script deve começar com o shebang:

```
#!/bin/bash
```

10.2 Variáveis

```
# Definindo variáveis
NOME='João'
IDADE=25
echo "Olá, $NOME! Você tem $IDADE anos."
```

10.3 Estruturas Condicionais

```
if [ $IDADE -ge 18 ]; then
echo 'Maior de idade'
else
echo 'Menor de idade'
fi
```

10.4 Loops

```
# Loop for
for i in {1..5}; do
echo "Número: $i"
done
```

```
# Loop while
contador=0
while [ $contador -lt 5 ]; do
echo "Contador: $contador"
contador=$((contador + 1))
done
```

10.5 Funções

```
saudar() {
```

```
echo "Olá, $1!"
}

saudar 'Maria'
```

10.6 Operadores de Comparação

Operador	Significado	Exemplo
`-eq`	Igual a	`[ $a -eq $b ]`
`-ne`	Diferente de	`[ $a -ne $b ]`
`-gt`	Maior que	`[ $a -gt $b ]`
`-lt`	Menor que	`[ $a -lt $b ]`
`-ge`	Maior ou igual	`[ $a -ge $b ]`
`-le`	Menor ou igual	`[ $a -le $b ]`
`==`	String igual	`[ "$str1" == "$str2" ]`
`!=`	String diferente	`[ "$str1" != "$str2" ]`
`-z`	String vazia	`[ -z "$str" ]`
`-n`	String não vazia	`[ -n "$str" ]`

10.7 Exemplo Prático

Script completo de backup:

```
#!/bin/bash
# Script de backup

DATA=$(date +%Y%m%d_%H%M%S)
ORIGEM='/home/usuario/documentos'
DESTINO='/backup'
ARQUIVO="backup_$DATA.tar.gz"

echo "Iniciando backup..."
tar -czf "$DESTINO/$ARQUIVO" "$ORIGEM"

if [ $? -eq 0 ]; then
echo "Backup concluído com sucesso!"
else
echo "Erro ao criar backup!"
```

```
exit 1
fi
```

Conclusão

Este guia apresentou os comandos essenciais do Linux, desde os básicos de navegação até conceitos intermediários como shell scripting. O domínio destes comandos é fundamental para qualquer profissional que trabalhe com sistemas Linux.

A prática constante é a chave para memorizar e dominar esses comandos. Experimente criar seus próprios scripts, automatizar tarefas do dia a dia e explorar cada comando em profundidade através de suas páginas de manual (`man comando`).

Próximos Passos

- Pratique regularmente os comandos apresentados
- Explore as páginas de manual (man) para aprofundar conhecimentos
- Crie scripts para automatizar tarefas repetitivas
- Configure seu próprio servidor Linux
- Aprenda sobre administração de sistemas e segurança
- Estude expressões regulares para melhorar uso de grep e sed
- Explore ferramentas avançadas como awk, perl e python

> ***Lembre-se:*** *O terminal Linux é uma ferramenta poderosa. Com prática e dedicação, você se tornará cada vez mais eficiente e produtivo!*