Etec@123

find

Encontrar arquivos pelo nome

find /etc -type f -name "nome_do_arquivo.txt"

Encontrar diretórios pelo nome

find /etc -type d -name "meudiretorio"

Encontrar arquivos com um padrão específico

find /etc -type f -name "*.png"

Encontrar arquivos por tamanho

find /etc -type f -size +2M

find /etc -type f -size -2M

Encontrar arquivos modificados no último dia

find /etc -type f -mtime -1

Encontrar e executar um comando nos arquivos

find /etc -type f -name "*.tmp" -exec rm {} \;

Encontrar arquivos com permissões específicas

find /etc -type f -perm 0644

Encontrar arquivos pelo proprietário

find /etc -type f -user nome_do_usuario

Encontrar arquivos pelo grupo

find /etc -type f -group nome_do_grupo

Encontrar arquivos e combinar condições

find /etc -type f \(-name "*.txt" -or -name "*.pdf" \)

Encontrar arquivos excluindo diretórios

find /etc -type f -name "*.log" ! -path "*/diretorio_para_excluir/*"

Limitar profundidade da busca

find /etc -maxdepth 2 -type f -name "*.jpg"

Encontrar links simbólicos

find /etc -type l

diff

Comparar dois arquivos

diff arquivo1.txt arquivo2.txt

Comparar dois arquivos, ignorando espaços em branco

diff -w arquivo1.txt arquivo2.txt

Comparar dois arquivos, mostrando as diferenças lado a lado

diff -y arquivo1.txt arquivo2.txt

Comparar dois arquivos, gerando saída em formato unificado (útil para patches)

diff -u arquivo1.txt arquivo2.txt

Comparar dois diretórios recursivamente

diff -r diretorio1 diretorio2

Comparar dois diretórios, mostrando apenas os nomes de arquivos diferentes

diff -rq diretorio1 diretorio2

Comparar dois arquivos, ignorando todas as linhas em branco

diff -B arquivo1.txt arquivo2.txt

Comparar dois arquivos, considerando maiúsculas e minúsculas irrelevantes

diff -i arquivo1.txt arquivo2.txt

tar

Criar um arquivo tar

tar -cvf arquivo.tar /caminho/para/diretorio

Criar um arquivo tar.gz (gzip)

tar -czvf arquivo.tar.gz /caminho/para/diretorio

Criar um arquivo tar.bz2 (bzip2)

tar -cjvf arquivo.tar.bz2 /caminho/para/diretorio

Listar o conteúdo de um arquivo tar

tar -tvf arquivo.tar

Extrair um arquivo tar

tar -xvf arquivo.tar

Extrair um arquivo tar.gz

tar -xzvf arquivo.tar.gz

Extrair um arquivo tar.bz2

tar -xjvf arquivo.tar.bz2

Extrair um arquivo tar em um diretório específico

tar -xvf arquivo.tar -C /caminho/para/diretorio

Criar um arquivo tar com exclusão de diretórios específicos

tar -cvf arquivo.tar --exclude='/caminho/para/excluir' /caminho/para/diretorio

Adicionar um arquivo a um arquivo tar existente

tar -rvf arquivo.tar novo_arquivo.txt

df

Exibir a utilização do espaço em disco de todos os sistemas de arquivos montados

df

Exibir a utilização do espaço em disco em megabytes

df -m

Exibir a utilização do espaço em disco em gigabytes

df -h

Exibir a utilização do espaço em disco com o sistema de arquivos do tipo

df -T

Exibir a utilização do espaço em disco de um sistema de arquivos específico

df /caminho/para/diretorio

Exibir informações usando o formato de saída POSIX

df -P

Exibir informações detalhadas de inodes (número de inodes livres e usados)

df -i

Exibir a utilização do espaço em disco para todos os sistemas de arquivos, excluindo os do tipo tmpfs e devtmpfs

df --exclude-type=tmpfs --exclude-type=devtmpfs

du

Exibir o uso do disco do diretório atual e seus subdiretórios em kilobytes

du

Exibir o uso do disco de um diretório específico

du /caminho/para/diretorio

Exibir o uso do disco em megabytes

du -m

Exibir o uso do disco em gigabytes

du -h

Exibir somente o total de uso do disco do diretório especificado

du -sh /caminho/para/diretorio

Exibir o uso do disco para todos os arquivos e diretórios dentro de um diretório específico

du -ah /caminho/para/diretorio

Exibir o uso do disco de todos os arquivos e diretórios até uma profundidade de 2 diretórios

du -h --max-depth=2 /caminho/para/diretorio

Listar o uso do disco de arquivos e diretórios e excluir aqueles que correspondem a um padrão específico

du -h --exclude='*.txt' /caminho/para/diretorio

Exibir o uso do disco de diretórios específicos com a soma total

du -csh /diretorio1 /diretorio2

useradd

Criar um novo usuário com o diretório home padrão

useradd nome_do_usuario

Criar um novo usuário com um diretório home especificado

useradd -d /caminho/para/home nome_do_usuario

Criar um novo usuário e especificar um shell personalizado

useradd -s /bin/zsh nome_do_usuario

Criar um novo usuário com um ID de usuário (UID) específico

useradd -u 1005 nome_do_usuario

Criar um novo usuário com uma descrição (comentário) como nome completo

useradd -c "João Silva" nome_do_usuario

Criar um novo usuário e adicionar ao grupo existente

useradd -G nome_do_grupo nome_do_usuario

Criar um novo usuário e definir a senha ao mesmo tempo (requer inserir a senha criptografada)

useradd -p senha_criptografada nome_do_usuario

Criar um novo usuário com um diretório home e criar o diretório se ele não existir

useradd -m nome_do_usuario

Criar um novo usuário e especificar múltiplos grupos iniciais

useradd -G grupo1,grupo2 nome_do_usuario

mount

umount

ps

Exibir todos os processos atuais no terminal

ps

Exibir todos os processos em execução como o usuário atual

ps x

Exibir todos os processos atuais, incluindo os que não estão associados a um terminal

ps aux

Exibir todos os processos atuais em formato de lista longa

ps -ef

Exibir todos os processos de um usuário específico

ps -u nome_do_usuario

Exibir processos com um formato específico

ps -eo pid,tt,user,fname,pmem

Exibir a árvore de processos para visualizar as relações entre eles

ps -ejH

Exibir processos com um cabeçalho, mas sem o cabeçalho padrão

ps --no-headers -e

Exibir os processos de um grupo específico

ps -g id_do_grupo

Exibir processos e ordenar pelo uso de CPU

ps --sort -pcpu

Exibir processos e ordenar pelo uso de memória

ps --sort -pmem

Exibir todos os processos associados a uma sessão específica

ps -s id_da_sessao

Atualizar a lista de processos a cada 2 segundos (útil para monitoramento)

watch -n 2 "ps aux"

kill

Enviar o sinal SIGTERM para um processo pelo seu PID (o padrão, para pedir ao processo para encerrar)

kill 1234

Enviar o sinal SIGKILL para um processo pelo seu PID (termina o processo imediatamente)

kill -9 1234

Enviar o sinal SIGKILL para vários processos ao mesmo tempo

kill -9 1234 5678 91011

Listar todos os sinais disponíveis que podem ser enviados aos processos

kill -l

Enviar um sinal específico pelo nome ao processo

kill -SIGTERM 1234

Enviar o sinal SIGHUP para todos os processos do grupo de processos atual

kill -HUP -1

Enviar o sinal SIGSTOP (pausar o processo) pelo seu PID

kill -SIGSTOP 1234

Enviar o sinal SIGCONT (continuar o processo pausado) pelo seu PID

kill -SIGCONT 1234

Enviar um sinal a processos por nome com o comando `pkill` (semelhante ao `kill`, mas pode usar nomes)

pkill -9 nome_do_processo

Enviar um sinal para todos os processos de um usuário específico

pkill -u nome_do_usuario

Codes

For

```
# Imprimir números de 1 a 5
for i in {1..5}
do
echo "Número: $i"
done

# Percorrer uma lista de nomes
for nome in Pedro Ana Carlos
do
echo "Nome: $nome"
done

# Executar um comando para cada arquivo com extensão .txt no diretório atual
for arquivo in *.txt
do
echo "Processando arquivo: $arquivo"
wc -l $arquivo  # Conta o número de linhas em cada arquivo
done
```

Utilizar o resultado de um comando como lista para o loop

for usuario in $(cat lista_usuarios.txt)

do

echo "Adicionando usuário: $usuario"

useradd $usuario

done

Percorrer arquivos e diretórios em um diretório específico

for item in /caminho/para/diretorio/*

do

if [-d "$item"]; then

echo "$item é um diretório"

elif [-f "$item"]; then

echo "$item é um arquivo"

fi

done

Usar a sintaxe C-like com o comando `for`

for ((i = 1; i <= 10; i++))

do

echo "Contando: $i"

done

While

Ler linhas de um arquivo até o fim

while read linha

do

echo "Linha: $linha"

done < arquivo.txt

-le = less or equal, menor ou igual

Contar de 1 até 5

contador=1

while [$contador -le 5]

do

echo "Contador: $contador"

contador=$((contador + 1))

done

##################################Paremo aqui

Executar um loop enquanto a condição de saída de um comando for zero (comando bem-sucedido)

while ping -c 1 192.168.1.1 &>/dev/null

do

echo "O host ainda está alcançável."

sleep 5

done

Utilizar um loop para esperar até que um arquivo seja criado

while [! -f /tmp/arquivo_esperado.txt]

do

echo "Aguardando a criação do arquivo..."

sleep 10

done

Um loop infinito que pode ser terminado com um comando interno ou interrupção externa

while true

do

echo "Pressione CTRL+C para sair."

sleep 1

done

Processar linhas de um arquivo e sair do loop com uma condição específica

while read linha

do

if [["$linha" == "sair"]]; then

break

fi

echo "Linha: $linha"

done < arquivo.txt

Esperar até que um serviço específico esteja rodando (simulado pelo comando grep)

while ! pgrep -x "nome_do_servico" >/dev/null

do

echo "Esperando o serviço iniciar..."

sleep 1

done

if

Testar se um arquivo existe

if [-f "/caminho/para/arquivo.txt"]; then

echo "O arquivo existe."

else

echo "O arquivo não existe."

fi

Testar se um diretório existe

if [-d "/caminho/para/diretorio"]; then

```
echo "O diretório existe."
else
echo "O diretório não existe."
fi

# Testar se uma variável é igual a um valor específico
variavel="Teste"
if [ "$variavel" = "Teste" ]; then
echo "A variável é igual a Teste."
else
echo "A variável não é igual a Teste."
fi

# Usar condições numéricas para comparar valores
numero=10
if [ $numero -eq 10 ]; then
echo "O número é igual a 10."
else
```

```
echo "O número não é igual a 10."
fi

# Combinar condições com AND
if [ -f "/caminho/para/arquivo1.txt" ] && [ -f "/caminho/para/arquivo2.txt" ]; then
echo "Ambos os arquivos existem."
else
echo "Um ou ambos os arquivos não existem."
fi

# Combinar condições com OR
if [ -f "/caminho/para/arquivo1.txt" ] || [ -f "/caminho/para/arquivo2.txt" ]; then
echo "Pelo menos um dos arquivos existe."
else
echo "Nenhum dos arquivos existe."
fi

# Verificar se uma variável está vazia
```

```
if [ -z "$variavelVazia" ]; then
echo "A variável está vazia."
else
echo "A variável não está vazia."
fi

# Verificar se uma variável não está vazia
if [ -n "$variavelCheia" ]; then
echo "A variável não está vazia."
else
echo "A variável está vazia."
fi

# Utilizar padrões avançados com regex
if [[ "exemplo.txt" =~ ^exemplo\.(txt|doc)$ ]]; then
echo "O nome do arquivo é 'exemplo.txt' ou 'exemplo.doc'."
else
echo "O nome do arquivo não corresponde."
```

```
fi

##########################################################
################

## Variaveis
##########################################################
########

##########################################################
################

# Definir uma variável
nome="João"
echo "O nome é $nome"

# Variáveis de ambiente (definir e usar)
export CAMINHO="/usr/local/bin"
echo "O caminho é $CAMINHO"

# Concatenar variáveis
primeiro_nome="Maria"
```

```
sobrenome="Silva"
nome_completo="$primeiro_nome $sobrenome"
echo "Nome completo: $nome_completo"

# Usar variáveis em cálculos aritméticos
a=5
b=3
soma=$((a + b))
echo "Soma: $soma"

# Incrementar e decrementar variáveis
contador=1
((contador++))
echo "Contador incrementado: $contador"
((contador--))
echo "Contador decrementado: $contador"

# Variáveis de string e manipulação
texto="Olá, mundo"
```

```
echo "Texto original: $texto"
texto_maiusculo=${texto^^}
echo "Texto em maiúsculas: $texto_maiusculo"
texto_minusculo=${texto,,}
echo "Texto em minúsculas: $texto_minusculo"

# Extrair substring
frase="Bem-vindo ao ChatGPT"
parte_da_frase=${frase:10:6}
echo "Parte extraída da frase: $parte_da_frase"

# Substituir parte de uma string
endereco="Rua Avenida, 123"
endereco_corrigido=${endereco/Rua/Avenida}
echo "Endereço corrigido: $endereco_corrigido"

# Usar variáveis para guardar o resultado de um comando
data_atual=$(date)
```

```
echo "Data atual: $data_atual"

# Passar variáveis para um script em uma subshell
var="valor"
resultado=$(bash -c "echo A variável é: $var")
echo "Resultado de subshell: $resultado"

# Verificar se uma variável está definida ou é nula
if [ -z "$nao_definida" ]; then
echo "Variável 'nao_definida' não está definida ou é nula."
else
echo "Variável 'nao_definida' tem valor."
fi

##########################################################
```

Funções

##
###########

##
#########################

Definir uma função simples

funcao_simples() {

echo "Esta é uma função simples."

}

funcao_simples # Chamando a função

Função com parâmetros

saudacao() {

echo "Olá, $1!" # $1 refere-se ao primeiro argumento passado para a função

}

saudacao "Maria" # Chamando a função com um argumento

```
# Função que retorna um valor
obter_soma() {
local a=$1
local b=$2
return $(($a + $b))
}
obter_soma 5 10
echo "A soma é: $?"  # $? captura o último valor de retorno

# Função para verificar a existência de um arquivo
verifica_arquivo() {
local arquivo=$1
if [ -f "$arquivo" ]; then
echo "O arquivo existe."
else
echo "O arquivo não existe."
fi
}
```

```
verifica_arquivo "/caminho/para/arquivo.txt"

# Função com retorno baseado em condição
eh_maior_que_10() {
if [ $1 -gt 10 ]; then
return 0  # Retorna verdadeiro em shell (sucesso)
else
return 1  # Retorna falso em shell (falha)
fi
}
eh_maior_que_10 15
if [ $? -eq 0 ]; then
echo "É maior que 10."
else
echo "Não é maior que 10."
fi

# Função com saída complexa
listar_arquivos() {
```

```
echo "Listando arquivos no diretório atual:"
ls
}
listar_arquivos

# Função recursiva
recursiva() {
local num=$1
if [ $num -le 0 ]; then
echo "Recursão completa."
else
echo "Nível: $num"
recursiva $(($num - 1))
fi
}
recursiva 5

# Armazenando a saída de uma função em uma variável
```

```
captura_saida() {
echo "Algum texto e números $((1 + 1))"
}
variavel=$(captura_saida)
echo "Capturado pela função: $variavel"

#!/bin/bash

# Solicitar ao usuário que digite números e armazenar a entrada
echo "Por favor, digite um número inicial:"
read numero_inicial
echo "Por favor, digite um número final:"
read numero_final

# Imprimir números do intervalo fornecido pelo usuário
for (( i=numero_inicial; i<=numero_final; i++ )); do
echo "Número: $i"
done
```

Solicitar ao usuário para entrar com três nomes

echo "Entre com três nomes, separados por espaço:"

read nome1 nome2 nome3

Percorrer a lista de nomes fornecida pelo usuário

for nome in $nome1 $nome2 $nome3; do

echo "Nome: $nome"

done

Solicitar ao usuário o caminho para verificar arquivos e diretórios

echo "Por favor, digite o caminho de um diretório para verificação:"

read caminho

Verificar se o diretório existe e percorrer arquivos e diretórios nele

```
if [ -d "$caminho" ]; then
for item in "$caminho"/*; do
if [ -d "$item" ]; then
echo "$item é um diretório"
elif [ -f "$item" ]; then
echo "$item é um arquivo"
fi
done
else
echo "O diretório fornecido não existe."
fi

# Solicitar ao usuário um valor para teste condicional
echo "Digite um número para teste:"
read numero_teste

# Testar se o número é igual a 10
if [ "$numero_teste" -eq 10 ]; then
```

```
echo "O número é igual a 10."
else
echo "O número não é igual a 10."
fi

# Definição de função que solicita um nome
saudacao() {
echo "Por favor, digite um nome para saudação:"
read nome
echo "Olá, $nome!"
}

# Chamar a função de saudação
saudacao

# Pedir ao usuário um valor para calcular a soma
echo "Digite dois números para calcular a soma:"
read a b
soma=$((a + b))
```

```
echo "Soma: $soma"


# As partes que dependem de condições externas (como verificar a rede ou esperar por arquivos) foram omitidas por serem mais complexas de simular interativamente.
```