

echoComandos comuns em terminais Windows e Linux:

Windows	Linux	Comando
Ctrl+c	Ctrl+c	Cancela uma ação no terminal.
TAB	TAB	Completa a digitação.
cls	clear	Limpa as informações na tela
cd	cd	Mudança de diretório.
md	mkdir	Criação de novo diretório.
rd	rm -r	Remoção de diretório
del	rm	Remoção de arquivo
copy	cp	Cópia de arquivo.
date	date	Mostra/altera data do sistema.
time	time	Mostra/altera hora do sistema.
move	mv	Move diretório/arquivo.
ren	mv	Renomear diretório/arquivo.
dir	ls	Exibe pastas do diretório atual
copy con	(vários, mas recomendo:) nano	Criação de arquivo.

AUTOMAÇÃO DE PROCESSOS E SOLUÇÕES

A partir desse momento veremos ferramentas que permitem a criação de soluções automatizadas dentro do ambiente Linux. Para isso, aprenderemos novos comandos e sintaxes no terminal.

COPIAR - MOVER - EXCLUIR

Essas são as ações básicas da manipulação de arquivos que precisaremos cobrir para auxiliar no uso dos demais comandos.

Comandos de cópia.

`cp /origem/arquivo /destino`

Cria uma cópia do arquivo da pasta origem dentro da pasta destino.

`cp -r /origem /destino`

Copia recursivamente todos os arquivos contidos na pasta origem na pasta destino.

`cp -i /origem /destino`

Copia os arquivos de forma interativa, perguntando se o usuário deseja substituir um arquivo já existente.

Comandos de exclusão

`rmdir /diretorio`

Remove por completo o diretório informado

- `rm -r` : remove os diretórios e arquivos dentro do caminho indicado (incluindo o diretório indicado)
- `rm -f`: remove arquivos somente leitura, sem pedir confirmação

Movendo arquivos

`mv /origem/arquivo /destino`

Semelhante ao `cp`, porém remove o arquivo da pasta de origem.

`mv /origem/arquivo /destino/novo-nome`

Movendo o arquivo, renomeando o mesmo na pasta destino.

CRIAÇÃO DE SCRIPTS

A criação de scripts no shell é uma das áreas de estudos mais interessantes ao estudarmos sistemas operacionais. Através deles que podemos automatizar ações repetitivas, assegurar que o correto passo-a-passo de uma tarefa seja realizado, etc.

Para criação de scripts, é ideal que se tenha conhecimento de uso de algum editor de textos no terminal. A princípio, utilizaremos o comando “`cat`” por ter uma sintaxe mais simples.

EXTENSÃO DE ARQUIVOS

Para que os scripts do shell sejam identificados automaticamente desse modo pelo linux, salvamos os mesmos com a extensão `.sh`.

ESTRUTURA DOS ARQUIVOS DE SCRIPT

Ao criar um novo script, a primeira tarefa é identificar qual shell será usado para realizar a execução do mesmo. Essa identificação é feita através da inserção do comentário:

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

Ou

```
#!/bin/SHELL_ATUAL
```

Você pode inserir comentários no seu script utilizando o caractere “`#`” antes do texto

INSERÇÃO DE COMANDOS NO SCRIPT

A partir desse momento fica a critério da criatividade do usuário e do seu conhecimento de comandos no terminal. Um script nada mais é que um compilado de comandos que podem ser executados em conjunto. Ainda é possível declarar variáveis e funções dentro de um script, mas isso será visto posteriormente.

PERMISSÃO DE EXECUÇÃO DO SCRIPT

O script precisa ter suas permissões modificadas para ser reconhecido como um executável no sistema. As permissões de arquivos são executadas com o comando “`chmod`”, e esse estudo também será visto com mais

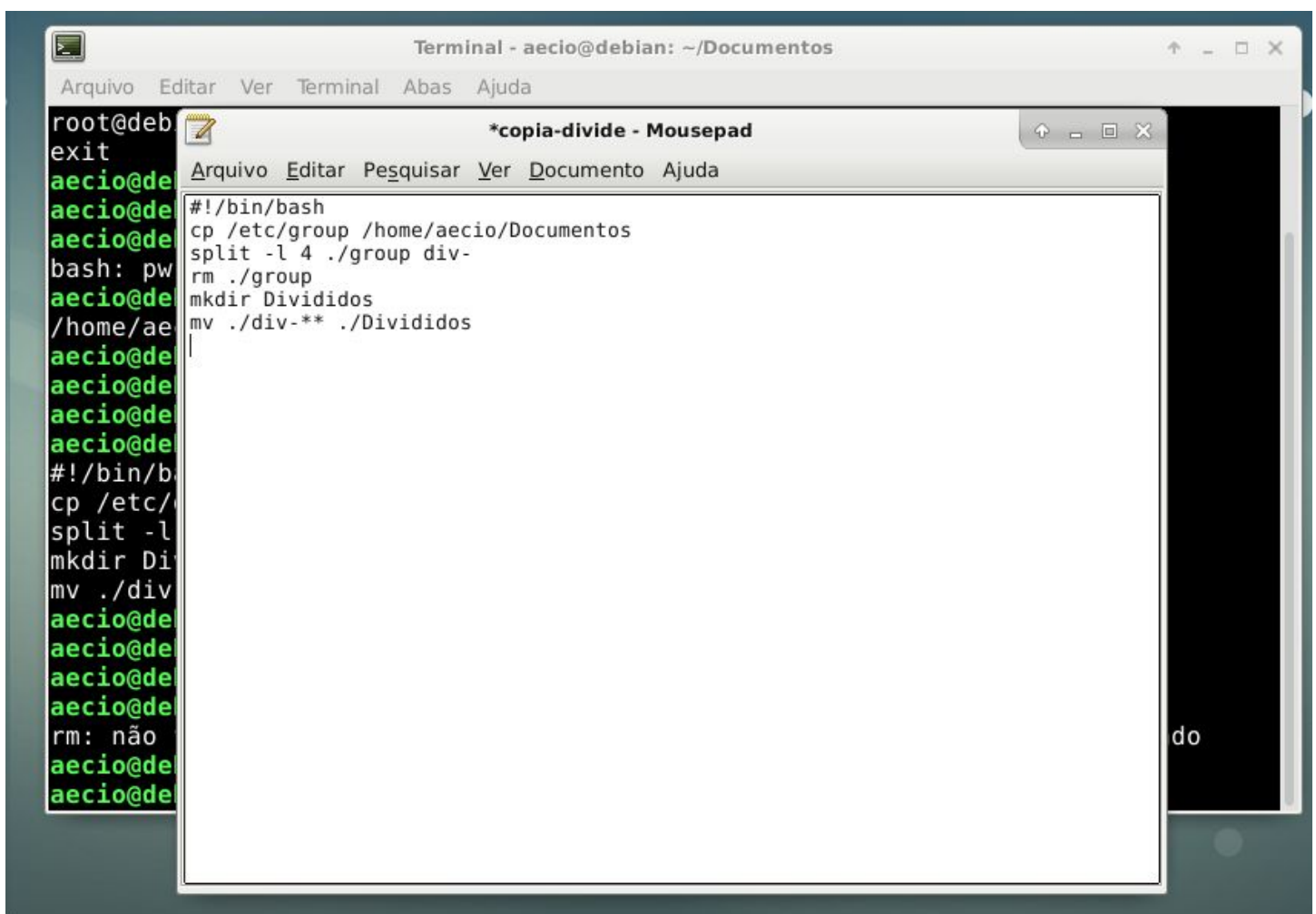
detalhes mais adiante. Por hora, use o comando “chmod a+x nomeDoScript.sh” para conceder permissões de execução ao mesmo.

Pronto, já é possível criar suas primeiras ações automatizadas no sistema.

Exemplo de problema:

- Copiar o arquivo “group” que se encontra no diretório “/etc/” para um novo diretório dentro da pasta “Documentos” que se encontra na sua pasta de usuário.
- Depois que esse documento estiver lá, o script deverá dividir o mesmo em pequenos documentos de 4 linhas.
- Feita a divisão, o script deve pegar todos os documentos criados (e apenas eles) e movê-los para um diretório criado ali mesmo, com o nome de “Divididos”.
- Por fim, o script deverá remover o arquivo “group” que foi copiado originalmente.

Solução:



The screenshot shows a terminal window titled "Terminal - aecio@debian: ~/Documentos" and a mousepad editor window titled "*copia-divide - Mousepad". The terminal window shows the following commands and output:

```
root@deb
exit
aecio@de
aecio@de
aecio@de
bash: pw
aecio@de
/home/ae
aecio@de
aecio@de
aecio@de
aecio@de
#!/bin/b
cp /etc/
split -l
mkdir Di
mv ./div
aecio@de
aecio@de
aecio@de
aecio@de
rm: não
aecio@de
aecio@de
```

The mousepad editor window shows the following script:

```
#!/bin/bash
cp /etc/group /home/aecio/Documentos
split -l 4 ./group div-
rm ./group
mkdir Divididos
mv ./div-* ./Divididos
```

VARIÁVEIS

A utilização de variáveis é o próximo passo para a nossa criação de scripts mais complexos. O conceito de variável utilizado no bash é bem próximo ao que já foi estudado em lógica de programação, são espaços de memória no qual armazenamos valores. Partiremos então para o entendimento prático de como criar variáveis, inserir e ler os dados das mesmas.

DECLARAÇÃO DE UMA VARIÁVEL

A declaração de uma variável no bash, diferente de outras linguagens de programação estudadas, não perguntam o tipo de dado a ser utilizado. Você deve apenas digitar o nome da variável. Para esse tipo de script, convencionou-se utilizar o nome da variável em maiúsculo.

Declara uma variável para receber um nome:
NOME

INSERÇÃO DE DADOS EM UMA VARIÁVEL

A inserção de valores em uma variável pode ser feita no momento da sua declaração. Para atribuir valores, utilizamos o sinal de igualdade, seguido do dado que iremos inserir. Note que não há espaços nessa operação:

```
NOME="Fulano de Tal"  
SITE="www.google.com"  
read NOME (o valor digitado pelo usuário será inserido na variável)
```

LEITURA DE DADOS DE UMA VARIÁVEL

Para acessarmos uma variável durante o script, utilizaremos o símbolo \$ para identificar a palavra como uma variável.

```
$NOME  
$SITE
```

EXIBIÇÃO DE DADOS

A escrita de dados na tela é uma rotina comum em scripts, seja para escrever uma mensagem ao usuário, mostrar valores de uma variável ou ambos. Exibimos dados em tela através do comando *echo*.

```
echo "Olá Mundo!"  
echo $SITE  
echo "Olá ${NOME}"  
echo "Olá ${NOME}zinho"
```

LEITURA DE DADOS

A leitura de dados é outro fator fundamental para interação com usuário. Ela geralmente é feita para que o usuário possa alimentar uma variável com um dado que será usado no script.

```
read NOME_VARIAVEL
```

Exercício:

1 - Crie um script que solicite do usuário o nome de 3 pastas e crie as mesmas no diretório atual.

2 - Crie um script que pergunte ao usuário qual o nome da pasta que ele deseja renomear. Em seguida, pergunte ao usuário qual o novo nome para a pasta. O script deverá renomear a pasta para o novo nome informado.

3- Crie um script que moverá todas as pastas do diretório atual (independente de qual diretório se encontra) para um diretório fixo e específico (um diretório de backup).

CONDICIONAIS

Controle de fluxo são comandos que vão testando algumas alternativas, e de acordo com essas alternativas, vão executando comandos. Um dos comandos de controle de fluxo mais usados é certamente o if, que é baseado na lógica “se acontecer isso, irei fazer isso, se não, irei fazer aquilo”.

- Sintaxe básica do comando:

```
if
then
    //comandos
elif
then
    //comandos
else
    //comandos
fi
```

- Exemplo:

```
if [ -e $linux ]
then
    echo 'A variável $linux existe.'
else
    echo 'A variável $linux não existe.'
fi
```

```
#!/bin/bash
if [ ! -e "/root/backup" ]
then
    mkdir /root/backup
    echo "Foi criada a pasta de backup"
else
    echo "Diretório ja existe"
fi
```

Exercício de Implementação: Se o diretório atual não for o diretório de backup, mover-se para o diretório de backup e lá dentro, listar todos os arquivos existentes em tela.

Em geral as linguagens de programação oferecem maneiras de verificar se um valor é igual, menor ou maior que outro. As formas mais comuns são:

- ==: Igual à
- != : Diferente de
- < : Menor que
- > : Maior que
- <= : Menor ou igual à
- >= : Maior ou igual à

Em shell script os mesmos testes são obtidos com:

- -eq : (equal) Igual à
- -ne : (not equal) Diferente de
- -lt : (less than) Menor que
- -gt : (greater than) Maior que
- -le : (less or equal) Menor ou igual à
- -ge : (greater or equal) Maior ou igual à

Ex:

```
#!/bin/bash
if [ 2 -eq 4 ]
then
    echo "Esse echo nunca acontecerá"
fi
```

As expressões acima são mais comuns para tratar números. Para fazer comparações com textos use:

- = : Igual à (isso mesmo apenas um sinal de igual)
- != : Diferente de
- -n : String existe e não é vazia (apenas um operador)
- **-z **: String existe e é vazia (apenas um operador)

Ex2:

```
#!/bin/bash
echo "Digite seu nome: "
read nome
if [ -z $nome ]
then
    echo "Você não digitou seu nome!"
else
    echo "Olá, $nome"
Fi
```

COMANDO CASE

O comando “case” é usado para executar um bloco de código de acordo com o valor de uma variável. O comando “case” é interessante pois pode definir diversas opções diferentes sem usar uma estrutura com diversos comandos “if”, “elif” e “else”. Veja um exemplo:

```
#!/bin/bash
echo -n "O que deseja fazer? (A)lmoçar/(J)antar/(S)air? [A] "
read resposta
case "$resposta" in
    a)
        echo "Então tenha um bom almoço =)"
        ;;
    j)
        echo "Um jantar vai muito bem."
        ;;
    s)
        echo "Saindo..."
        ;;
    *)
        echo "Opção inválida"
        ;;
esac
```

Exercício: Crie um jogo do tipo Quiz (perguntas e respostas) que deve conter ao menos 5 perguntas de múltipla escolha, com 4 opções em cada pergunta (a, b, c, d). O jogo deve perguntar o nome do jogador e no final exibir uma mensagem parabenizando o mesmo pelo resultado, se o jogador acertou as 5 respostas, ou mostrar a pontuação total caso tenha feito de 1 a 4, ou ainda pedir pra tentar novamente, caso tenha feito 0.

REPETIÇÃO

Uma estrutura de repetição, também conhecida como loop ou laço, é uma estrutura de desvio do fluxo de controle presente em linguagens de programação que realiza e/ou repete diferentes algoritmos/ações dependendo se uma condição é verdadeira ou falsa.

FOR

O laço for executa uma ação até que uma condição seja atendida. Existem outras sintaxes e diferentes formas de utilizar o FOR, mas utilizaremos essa por ser mais próxima das linguagens de programação já estudadas. Sintaxe:

```
for [ condição ];
do
#Seu codigo
done
```

Ex:

```
#!/bin/bash
echo "Contando ate 5 la estilo C"
for (( i=0; i<=5; i++))
```

```
do
    echo "Executando o $i"
done
```

- Atividade For: Crie um script que pergunte ao usuário a quantidade de pastas que ele deseja criar. Em seguida, repita a criação das pastas quantas vezes o usuário informou, perguntando o nome da pasta a ser criada.

WHILE

O laço for executa uma ação até que uma condição seja atendida.

Sintaxe:

```
while [ condição ];
do
#Seu codigo
done
```

Ex:

```
#!/bin/bash
contador=1
while [ $contador -ne 4 ];
do
    echo "Mensagem exibida $contador vez".
    ((contador=$contador+1))
done
```

Ex2:

```
#!/bin/bash
while [ $RESPOSTA -ne "Salvador" ];
do
    echo "Qual a capital de Bahia?".
    read RESPOSTA
done
```

AUTOMAÇÃO PRÁTICA

<https://www.diolinux.com.br/2017/06/shell-script-instalar-programa-linux.html>

GERENCIAMENTO DE USUÁRIOS

No Linux nós podemos dividir os usuários em 3 tipos:

- Usuários Comuns
 - Em geral, podem manipular arquivos em seu diretório e em outros diretórios. Porém, existem restrições de arquivos que podem acessar e funções que podem executar.
- Usuários de Sistema

- Não são usuários conectáveis. O sistema utiliza esses usuários para propósitos específicos.
- Temos como exemplo:
 - Nobody
 - ip
- Root
 - Superusuário com acesso irrestrito ao sistema. Algumas ações e funcionalidades do sistema são exclusivas do usuário root.

ADICIONANDO USUÁRIOS

As ações a seguir geralmente podem ser executadas em modo gráfico. Utilizaremos o terminal por ser um meio um pouco mais universal de uso do sistema.

Comando para adição de um novo usuário:

```
sudo adduser "nomeusuario"
```

Após a inserção do comando, o sistema solicitará dados adicionais como a senha para o usuário, o nome completo e demais informações de cadastro.

REMOVENDO USUÁRIOS

O comando para remoção de usuários via terminal é:

```
sudo deluser "nomeusuario"
```

Após a realização da ação, todos os diretórios e configuração referentes a esse usuário serão perdidas do sistema.

ALTERAÇÃO DE SENHA DO USUÁRIO

Para alterarmos a senha de um usuário, utilizaremos o seguinte comando no terminal:

```
sudo passwd "nomeusuario"
```

Será solicitado a inserção e confirmação da nova senha, e caso todo o processo seja feito corretamente o sistema informará uma mensagem confirmando a realização da tarefa.

LOGIN E LOGOUT DE USUÁRIO

Para fazer login via terminal no novo usuário do sistema, utilize o comando:

```
sudo login
```

Após o comando, será solicitado o nome e senha do usuário no qual pretende realizar o login.

O logout pode ser feito através do comando:

```
sudo logout
```

