Data Engineering

Filipe Nascimento
fgvn@cesar.school

C.E.S.A.R

C . E . S . A . R

Pessoas impulsionando inovação.

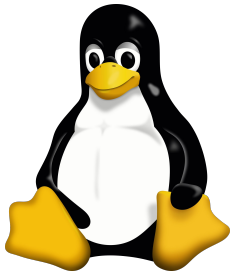
Inovação impulsionando negócios.

Everton Dias
etgdb@cesar.org.br

Janaína Branco
jcb@cesar.org.br

SHELL SCRIPT

Seguindo nossos estudos sobre o S.O Linux hoje abordaremos de maneira prática o Shell Script como ferramenta de automação de tarefas.



SHELL SCRIPT

SHELL SCRIPT

Utilizando o interpretador de comandos do Linux, o shell ou bash, é possível criarmos scripts que realizam tarefas desde as mais simples até as mais complexas dentro de nosso sistema operacional.

Para isso é importante entender alguns conceitos básicos como a utilização de comandos shell, criação de arquivos e manipulação de permissões.

Na aula de hoje de maneira prática começaremos a montar nossos primeiros scripts utilizando bash de nossos ambientes Linux.

PRÁTICA

LINUX

ATIVIDADE PRÁTICA 1

1. Utilizando sua linha de comando, navegue até a pasta do seu usuário
    a. Crie uma pasta chamada Aula03;
    b. Entre na pasta Aula03;
    c. Crie um arquivo chamado OlaMundo.sh utilizando os comandos touch ou algum outro que prefira
    d. Abra o arquivo no seu editor de textos preferido;
    e. No cabeçalho deste arquivo digite a linha de comando necessário para que o interpretador saiba que ali se trata de um script bash;
    f. Digite um comando que escreva "Olá Mundo!" no console ;
    g. Feche o editor de texto salvando seu arquivo;
    h. Torne seu arquivo executável;
    i. Execute-o salvando o seu conteúdo de saída dentro de um arquivo neste mesmo diretório chamado outputex1.txt;

LINUX

ATIVIDADE PRÁTICA 2

1. Ainda na Pasta criada no exercício anterior crie um script que:
    a. Escreva no console o que estará fazendo em cada etapa;
    b. Na primeira Etapa ele irá criar dois arquivos com o nome de log1.txt e log2.txt
    c. Irá armazenar o conteúdo do comando ls-lha dentro do log1.txt
    d. Irá armazenar o conteúdo do comando top dentro do log2.txt
    e. Irá criar uma pasta chamada backup_logs
    f. Realize uma cópia destes dois arquivos para dentro da pasta backup_logs
    g. Ao término da execução seu script deve informar que concluiu a atividade com sucesso;
    h. Torne-o executável;
    i. Rode seu script.

COMANDOS

LINUX

ESTRUTURA BÁSICA

```
#!/usr/bin/env bash

NOME="CESAR"
  echo "Olá $NOME!"
```

LINUX

VARIÁVEIS

```
ESCOLA=”CESAR”
echo $ESCOLA
echo “$ESCOLA”
echo “${ESCOLA}!”
echo “Estudo no ${ESCOLA}”
```

LINUX

ESTRUTURAS DE ITERAÇÃO

```
for i in /etc/rc.*; do
  echo $i
done
```

Iteração Básica

```
for ((k = 0 ; k < 100 ; k++)); do
  echo $k
done
```

Iteração C-Like

LINUX

ESTRUTURAS DE ITERAÇÃO

```
cat file.txt | while
read line; do
  echo $line
done
```

Iteração por Linha

```
while true; do
   ...
done
```

Iteração pela Eternidade

LINUX

ESTRUTURAS DE ITERAÇÃO

```
for i in {1..5}; do
    echo "Olá $i"
done
```

Ranges

LINUX

CONDICIONAL

```
if [[ -z "$string" ]]; then
  echo "STRING VAZIA!!"
elif [[ -n "$string" ]]; then
  echo "$string"
else
  echo "Não acontecerá!"
fi
```

LINUX

CONDICIONAL

`[[ -z STRING ]]`	String Vazia
`[[ -n STRING ]]`	String Não Vazia
`[[ STRING == STRING ]]`	Igualdade
`[[ STRING != STRING ]]`	Diferente (¬ Igualdade)

LINUX

CONDICIONAL

`[[ NUM -eq NUM ]]`	Igualdade
`[[ NUM -ne NUM ]]`	Diferente
`[[ NUM -lt NUM ]]`	Menor que
`[[ NUM -le NUM ]]`	Menor que ou Igual

LINUX

CONDICIONAL

`[[ NUM -gt NUM ]]`	Maior que
`[[ NUM -ge NUM ]]`	Maior que ou igual
`[[ STRING =~ STRING ]]`	Regexp
`(( NUM < NUM ))`	Condicional numérica

LINUX

CONDICIONAL PARA ARQUIVOS

`[[ -e FILE ]]`	Exists
`[[ -r FILE ]]`	Readable
`[[ -h FILE ]]`	Symlink
`[[ -d FILE ]]`	Directory

LINUX

CONDICIONAL PARA ARQUIVOS

`[[ -w FILE ]]`	Writable
`[[ -s FILE ]]`	Size is > 0 bytes
`[[ -f FILE ]]`	File
`[[ -x FILE ]]`	Executable

LINUX

CONDICIONAL PARA ARQUIVOS

`[[ FILE1 -nt FILE2 ]]`	FILE1 É MAIS RECENTE QUE FILE2
`[[ FILE1 -ot FILE2 ]]`	FILE2 É MAIS RECENTE QUE FILE1
`[[ FILE1 -ef FILE2 ]]`	FILE2 É MAIS RECENTE QUE FILE1

PRÁTICA

LINUX

ATIVIDADE PRÁTICA 3

1. Ainda na Pasta criada no exercício anterior crie um script que:
    a. Escreva no console o que estará fazendo em cada etapa;
    b. Você deve criar um script que registre uma quantidade informada ao invocar o script e que seja maior que 2, em caso de número menor de iterações informar que não será necessário e terminar a execução, o parâmetro de entrada determinará o número de iterações do comando top e para que seja salvo em um arquivo que respeitando a nomenclatura iteracao-<numero_da_iteração>.log
    c. Ao término replique os arquivos de iteração ímpar para uma pasta chamada backup_impar e os de numeração par para uma pasta chamada backup_par;
    d. Após essa cópia junte as duas pastas em um arquivo tarball chamado backup_par_impar
    e. Faça commit todas as atividades de hoje para o repositório do time em uma pasta com seu nome;

PRÁTICA

LINUX

ATIVIDADE PRÁTICA 4

1. A empresa está reclamando bastante do consumo de alguns processos nos servidores de bancos de dados que você está gerenciando - desenvolva um script que registre em log os 5 processos que estão consumindo mais memória e salve no seguinte formato as informações:
    a. AAAA-MM-DD, hh:mm:ss, PROCESSO , xxx MB - onde:
        i. AAAA - ano com 4 dígitos
        ii. MM - mês com 2 dígitos
        iii. DD - dia com 2 dígitos
        iv. hh:mm:ss - hora, minutos e segundos;
        v. xxx MB - quantidade de recursos consumidos em Megabytes

GERENCIADOR DE PACOTES

GERENCIADOR DE PACOTES

Definido como um conjunto de ferramentas que realizam a gestão de pacotes, sendo responsável por automatizar o processo de instalação, atualização, configuração e remoção de programas dentro do sistema operacional Linux que por sua vez herdou do Unix.

Dentre suas funções estão:

- Verificação da consistência dos pacotes baixados através do checksum;
- Verificar a assinatura digital para autenticar a origem de cada pacote;
- Atualizações e correções de bug abordando principalmente o aspecto da segurança;
- Gerenciar dependências para que ter certeza de que um pacote só será instalador se todas as dependências estiverem satisfeitas - dependency hell;

GERENCIADOR DE PACOTES

PRÁTICA

CRONTAB

O Cron é um recurso de agendamento de trabalhos baseado em tempo disponível em plataformas do tipo Unix, o que por sua vez inclui os sistemas operacionais Linux. Atividades registradas dentro desse recurso são executadas em segundo plano, sendo chamadas de “CRON JOBS”, sendo bastante útil para realizar tarefas relacionadas a manutenção do dia a dia do engenheiro de dados.

CRONTAB

Instalando o CRON

1. sudo apt update
2. sudo apt install cron
3. sudo systemctl enable cron

CRONTAB

Os cronjobs registrados ficam salvos em um registro e são geridos através de um arquivo especial chamado de crontab. Cada perfil no sistema pode ter seu próprio arquivo crontab, onde o usuário poderá agendar suas tarefas que ficará armazenado em /var/spool/cron/crontabs/

Para registrar um trabalho no agendamento, basta abrir o arquivo do crontab para realizar a edição e adicionar a tarefa em forma de uma expressão cron.

CRONTAB

SINTAXE:

minute hour day_of_month month day_of_week command_to_run

minuto	0-59
hora	0-23
Dia do mês	1-31
mês	1-12 ou JAN-DEC
Dia da semana	0-6 ou SUN-SAT

CRONTAB

Parâmetros a serem utilizado no campo referente as datas de agendamento:

- `* * * * *` - Executar o comando a cada minuto.
- `12 * * * *` - Executar o comando 12 minutos após cada hora.
- `0,15,30,45 * * * *` - Executar o comando a cada 15 minutos.
- `*/15 * * * *` - Executar o comando a cada 15 minutos.
- `0 4 * * *` - Executar o comando todo dia às 4:00 AM.
- `0 4 * * 2-4` - Executar o comando toda terça-feira, quarta-feira e quinta-feira às 4:00 AM.
- `20,40 */8 * 7-12 *` - Executar o comando no vigésimo e quadragésimo minuto de cada oitava hora, todos os dias dos últimos 6 meses do ano.

CRONTAB

EXEMPLO:

30 18 * * * echo ‘Hora do café’ >> /home/cesar/log.txt

CRONTAB

EXEMPLO:

30 18 * * * echo ‘Hora do café’ >> /home/cesar/log.txt

Execução todos os dias às 18h30!

PRÁTICA

LINUX

ATIVIDADE PRÁTICA 5

1. Desenvolva um script que escreva em um log separado por horário, os processos que mais estejam consumindo recursos do Sistema Operacional, registrando sua execução em um crontab para que rode a cada 10 minutos durante o dia;
2. Crie um script que realize o download do repositório deste link , copie o conteúdo referente aos datasets owid-covid-data.csv e owid-covid-codebook.csv renomeando-os para dados-covid-owid.csv e manual-dados-covid-owid.csv;
3. [DESAFIO] Crie um script que leia os dados do arquivo dados-covid-owid.csv, extraia apenas os países e ordene em ordem alfabética salvando em um novo arquivo chamado paises-afetados-covid-owid.txt;;

DÚVIDAS