CURSO DE LINUX

Aula 6

SSH e Scripts

Jorge Eduardo e Enrico Manfron
petcocelinux@gmail.com

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

PET-CoCE

DAINF

REVISÃO

CONDIÇÕES DO IF

➔ Para as condições dentro do podemos usar parâmetros comuns:

- [**n** string1]: o comprimento de string1 é diferente de 0;
- [**z** string1]: o comprimento de string1 é zero;
- [string1 **=** string2]: string1 e string2 são idênticas;
- [string1 **!=** string2]: string1 e string2 são diferentes;

CONDIÇÕES DO IF

➔ Para as condições dentro do podemos usar parâmetros comuns:

- [inteiro1 **-eq** inteiro2]: inteiro1 possui o mesmo valor que inteiro2;
- [inteiro1 **-ne** inteiro2]: inteiro1 não possui o mesmo valor que inteiro2;
- [inteiro1 **-gt** inteiro2]: inteiro1 é maior que inteiro2;
- [inteiro1 **-ge** inteiro2]: inteiro1 é maior ou igual a inteiro2;
- [inteiro1 **-lt** inteiro2]: inteiro1 é menor que inteiro2;
- [inteiro1 **-le** inteiro2]: inteiro1 é menor ou igual a inteiro2;

CONDIÇÕES DO IF

➔ Para as condições dentro do podemos usar parâmetros comuns:

- [**e** nome_do_arquivo]: verifica se nome_do_arquivo existe;
- [**d** nome_do_arquivo]: verifica se nome_do_arquivo é um diretório;
- [**f** nome_do_arquivo]: verifica se nome_do_arquivo é um arquivo regular (texto, imagem, programa, docs, planilhas).

ARGUMENTOS

➔ Normalmente um programa recebe argumentos como entrada, igual aos comandos.

➔ Em shell script não poderia ser diferente:

- $0 - contém o nome do script que foi executado
- $1 até $n - contém os argumentos na ordem em que foram passados
- $# - contém o número de argumentos que foi passado
- $* - retorna todos os argumentos

SHELL SCRIPT VARIÁVEIS

SHELL SCRIPT

VARIÁVEIS ESPECIAIS:

- **$ $$**
  - guarda a pid do script atual
- **$ $!**
  - guarda a pid do último job em segundo plano
- **$ ...**
  - parâmetro número n
- **$ ${VAR}**
  - mesmo que $var, só que sem conflitos
- **$ ${#VAR}**
  - retorna o tamanho da palavra

SHELL SCRIPT

OPERADORES:

- → $ +
  - Adição
- → $ -
  - subtração
- → $ *
  - Multiplicação

SHELL SCRIPT

OPERADORES:

- → $ /
  - Divisão
- → $ %
  - resto
- → $ **
  - Exponenciação

Ex: $((2**10))

SHELL SCRIPT

VARIÁVEIS ESPECIAIS:

- **$ $0**
  - Guarda o nome do programa .
- **$ ${1} ATÉ ${100..}**
  - Variáveis que guardam Parâmetros.
- **$ $#**
  - Número de parâmetros.
- **$ $***
  - Todos os Parâmetros, como palavra única.

SHELL SCRIPT LOOPS

SHELL SCRIPT

LOOPS CONDICIONAIS (FOR):

➔ for ((i = 0; i < 10; i++)); do
    Ações
done

➔ for i in VALOR_1 VALOR_2 ... VALOR_N;
    do
        AÇÕES
    done

SHELL SCRIPT

LOOPS CONDICIONAIS (FOR):

➔ usar {10..0} é como escrever {10,9,8,7,6...}

➔ for i in {10..0};
do
echo “$i”
done

SHELL SCRIPT

COMANDO:

- **$ SEQ <LIMITE_1> <SALTO> <LIMITE_2>**
  - Imprime a mensagem na tela uma sequencia de numeros desde limite um até limite 2 pulando n números de acordo com o salto.

- EXEMPLO:

```
for i in $(seq 1 5 100);
do
    echo “$i”
done
```

SHELL SCRIPT

LOOPS CONDICIONAIS (FOR):

➔ E para parametros podemos usar argumentos:

➔ for VARIAVEL in $*;
do
AÇÕES
done

SHELL SCRIPT

LOOPS CONDICIONAIS (WHILLE):

➔ Ele é bom geralmente quando não sabemos até onde contar :

➔ while [CONDICAO];
do
AÇÕES
done

SHELL SCRIPT

LOOPS CONDICIONAIS (WHILLE):

➔ EXERCÍCIO : faça um script que só fecha quando você digitar -1 .

```
#!/bin/bash
 echo “Informe o que você quiser, -1 para sair”
 read dado;
 while [ $dado != “-1” ];
 do
  echo “Você digitou $dado”
  read dado;
 done
```

SHELL SCRIPT

FUNÇÕES:

- O uso de funções se faz extremamente necessário na hora de separar e organizar seu código:
- nome_funcao()

```
{
    AÇÕES
}
```

SHELL SCRIPT

/EXERCÍCIO:

- ➔ Não digite **:(){ :|:& };:** e aperte enter.
- ➔ O que o trecho azul acima está fazendo?
- ➔ Faça um script que tenha uma função que imprime na tela "eae bro". E chame ela em seu script.

SSH

SSH - SECURE SHELL

➔ PROTOCOLO SSH

- Usado para conexão remota e segura em redes inseguras
- Criptografia de ponta a ponta

➔ SSH1

- Vulnerável e obsoleto

➔ SSH2

- Estável e seguro

➔ WINDOWS 10

- Cliente nativo SSH

SSH - SECURE SHELL

- **SUDO APT-GET INSTALL OPENSSH-CLIENT**
  - Instala o cliente OpenSSH no linux
- **SUDO APT-GET INSTALL OPENSSH-SERVER**
  - Instala o servidor no linux

SSH - SECURE SHELL

- **SSH-KEYGEN**
  - Gera um par de chaves
- **ID_RSA**
  - Chave **privada** de identificação
- **ID_RSA.PUB**
  - Chave **pública** de identificação
- **AUTHORIZED_KEYS**
  - Arquivo necessário para configuração de segurança
- **KNOWN_HOSTS**
  - Arquivo necessário para configuração de segurança

SSH - SECURE SHELL

- **SSH <USUÁRIO>@<IP> -X**
  - Acessa remotamente o servidor
- **SCP -R <DIRETÓRIO> <USUÁRIO>@<IP>:<DESTINO>**
  - Copia um diretório de forma recursiva do cliente para um destino específico no servidor
- **SCP -R <USUÁRIO>@<IP>:<DIRETÓRIO> <DESTINO>**
  - Copia um diretório de forma recursiva do servidor para um destino específico no cliente

SSH - SECURE SHELL

- **SCREEN (ENTER DUAS VEZES)**
  - Ativa a função screen na conexão remota
- **CRTL+A+D**
  - Coloca o screen em segundo plano (“background”)

SSH - SECURE SHELL

- **SCREEN -RD PID**
  - Coloca o screen em primeiro plano (“foreground”)
- **EXIT (NO SCREEN)**
  - Finaliza o screen
- **SCREEN -LIST**
  - Lista todos os screens ativos

CANIVETE SUIÇO DO SHELL

SSH - SECURE SHELL

https://www.brazilfw.com.br/downloads/users_old/glauber/guia_shell/canivete.html

ESTA FOI A AULA 6!