



BASH
THE BOURNE-AGAIN SHELL

Shell Script

Prof. Vinícius Vieira

Sumário

- Introdução
- I – Conceitos
- II – Variáveis e caracteres úteis
- III – Parâmetros
- IV – Estrutura de Controle de Fluxo
- V – Tratamento de Conteúdo
- VI – Expressões Regulares (regex)
- VII – Funções
- Conclusão



Introdução

Por que eu
preciso
aprender
Shell Script?



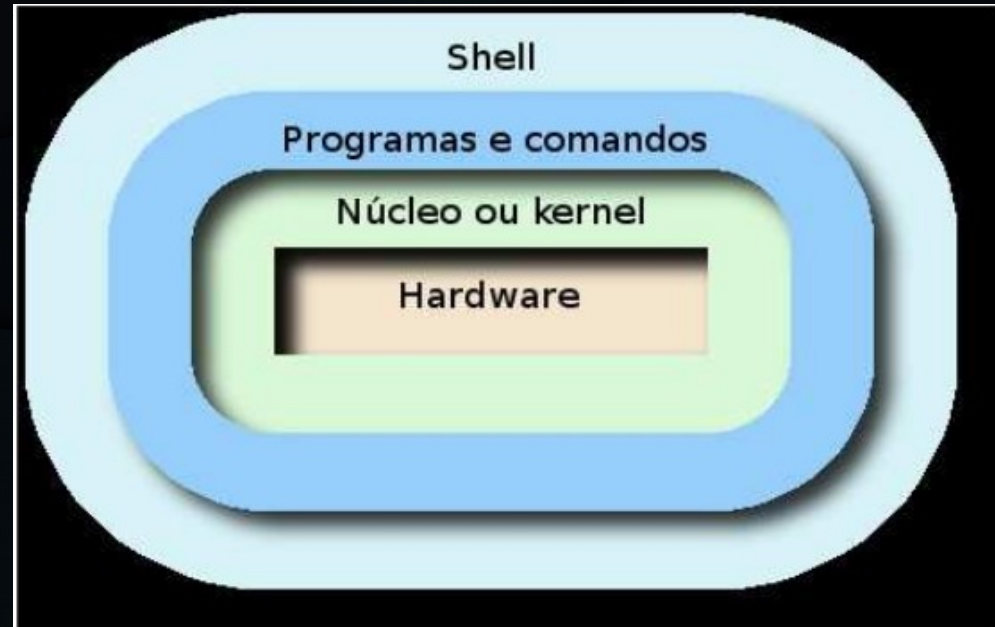
Introdução



MAY THE
BASH
BE WITH
YOU

I - Conceitos

- O que é Shell?
- Bash, sh, ksh
- e Script?



I - Conceitos

```
user@kali:~# ./programa [parâmetro] > destino
```


I - Conceitos

Permissões

```
user@kali:~# ls -l
```

```
-rwxr-xr-x 2 root root 4096 jun 15 17:49 arquivo.txt
```

| Primeiro caractere | Cada trinca de caractere | Significado dos caracteres |
|---|---|---|
| d => diretório b => arquivo de bloco c => arquivo especial de caractere p => canal s => socket - => arquivo "normal" | rw- => a primeira parte significa permissões do proprietário rw- => a segunda parte significa permissões do grupo ao qual o usuário pertence r-- => a terceira parte significa permissões para os demais usuários | r => significa permissão de leitura (read); w => significa permissão de gravação (write); x => significa permissão de execução (execution); - => significa permissão desabilitada. |

I - Conceitos

Definindo Permissões

user@kali:~# **chmod** [argumentos] [arquivo ou diretório]

| QUEM | | O QUÊ |
|--------------|----------------------------------|---------------|
| u => usuário | + (sinal de adição) => adicionar | r => leitura |
| g => grupo | permissão | w => gravação |
| o => outro | - (sinal de subtração) => | x => execução |
| a => todos | remover permissão | |
| | = (sinal de igualdade) => | |
| | definir permissão | |

I - Conceitos

Definindo Permissões

| Permissão | Binário | Decimal |
|-----------|---------|---------|
| --- | 000 | 0 |
| --x | 001 | 1 |
| -w- | 010 | 2 |
| -wx | 011 | 3 |
| r-- | 100 | 4 |
| r-x | 101 | 5 |
| rw- | 110 | 6 |
| rwX | 111 | 7 |

| | |
|-----------|-----|
| ----- | 000 |
| r----- | 400 |
| r--r--r-- | 444 |
| rw----- | 600 |
| rw-r--r-- | 644 |
| rw-rw-rw- | 666 |
| rwX----- | 700 |
| rwXr-x--- | 750 |
| rwXr-xr-x | 755 |
| rwXrwXrwX | 777 |

I - Conceitos

Definindo Permissões

```
# chmod 644 arquivo.txt  
# chmod rwrr arquivo.txt  
  
# chmod 777 arquivo.txt  
# chmod rwxrwxrwx arquivo.txt
```

r = 4

w = 2

x = 1

Exercício 1

Faça um Shell Script que, ao ser executado:

- Imprima na tela a frase “Bom dia usuário”
- em seguida, faça um ping no site www.google.com
- em seguida, imprima na tela “O site está funcionando”

II – Variáveis e Caracteres Úteis

```
shell@kali:~$ ze="Mesut Özil Backpfeifengesicht"  
shell@kali:~$ echo $ze  
Mesut Özil Backpfeifengesicht
```



II – Variáveis e Caracteres Úteis

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

```
echo "Digite um número"  
read num
```

```
echo "O número foi $num"
```

```
user@kali:~# ./script1.sh
```

```
Digite um número  
7
```

```
O número foi 7
```


II – Variáveis e Caracteres Úteis

Caracteres Úteis

Aspas “ - Tudo entre aspas é ignorado, exceto \$ ` \

Crase ` - Executa um comando entre crases

Barra invertida \ - Ignora apenas o caractere seguinte

Apóstrofo ‘ - Ignora tudo entre apóstrofes.

Ponto e vírgula ; - Mudança de linha

II – Variáveis e Caracteres Úteis

Redirecionamentos

`stdout` – saída padrão, seu default é a tela

`stdin` – entrada padrão, seu default é o teclado

`stderr` – saída de erro, seu default também é a tela

II – Variáveis e Caracteres Úteis

Redirecionamentos

- > - redirecionamento de saída (sobrescreve)
- >> - redirecionamento de saída (incrementa)
- 2> - redirecionamento de erro
- < - redirecionamento de entrada
- | - concatena uma saída em uma entrada

II – Variáveis e Caracteres Úteis

Expressões Aritméticas (expr)

expr 2 * 7

$$2 \times 7 = 14$$

expr 2 + 2

$$2 + 2 = 4$$

expr 2 / 2

$$2 \div 2 = 1$$

expr 2 - 2

$$2 - 2 = 0$$

II – Variáveis e Caracteres Úteis

Expressões Aritméticas (echo)

```
echo $(((2+3)*5))
```

```
tres=3
```

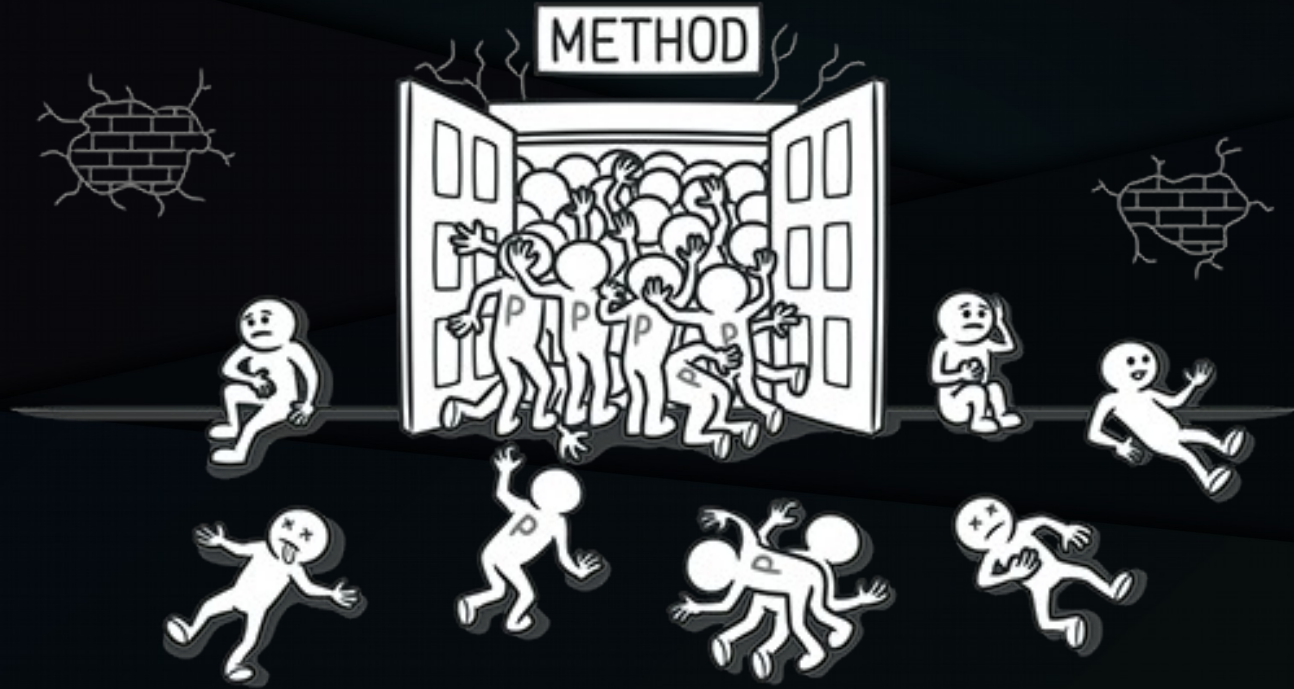
```
echo $((2+tres)*5))
```


Exercício 2

Faça um shell script que:

- Imprima na tela a saudação “Bom dia usuário” (o usuário tem que ser o usuário corrente)
- Imprima na tela a frase “Qual o site desejado” (o usuário irá passar algum site)
- Em seguida faça um ping para o site e salve o seu conteúdo no arquivo ping.txt (nenhum dado pode ser exibido na tela)
- Por fim, deverá aparecer “você está no diretório X e o arquivo com o resultado do ping é Y”

III – Parâmetros



III – Parâmetros

```
user@kali:~# ./programa [parâmetro1] [parâmetro2]
```

\$0

\$1

\$2

III – Parâmetros

\$0 ↔ **\$9** - Parâmetros possíveis

\${10} - Referencia o parâmetro 10

\$* - Utz todos parâmetros usados como string única

\$? - Saber se foi executado com sucesso [1 ou 0]

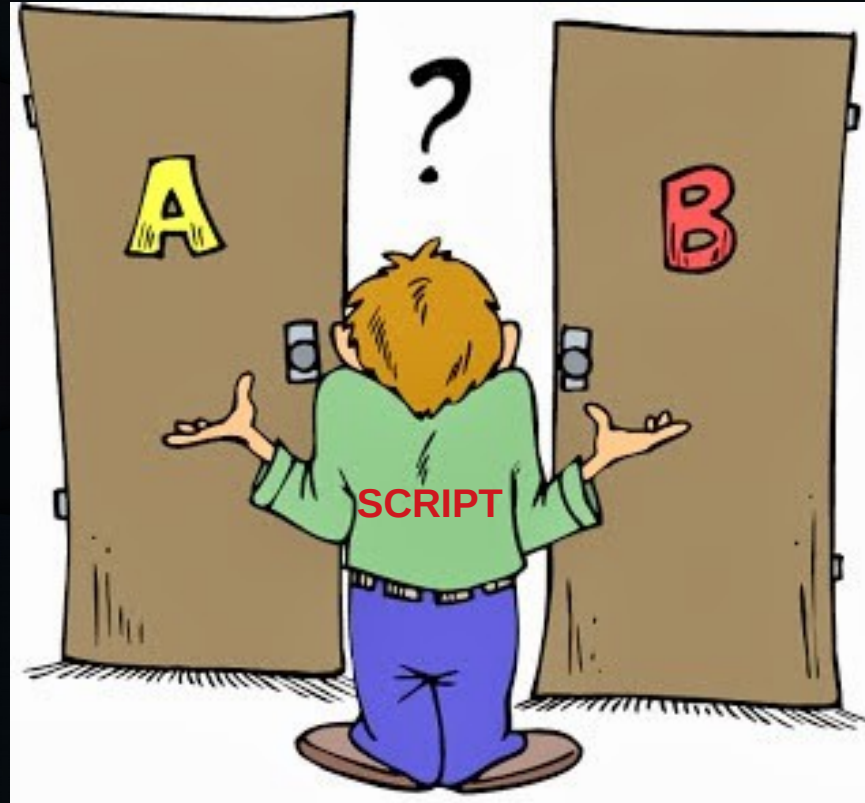
\$# - Saber qtd parâmetros inseridos pelo usuário

Exercício 3

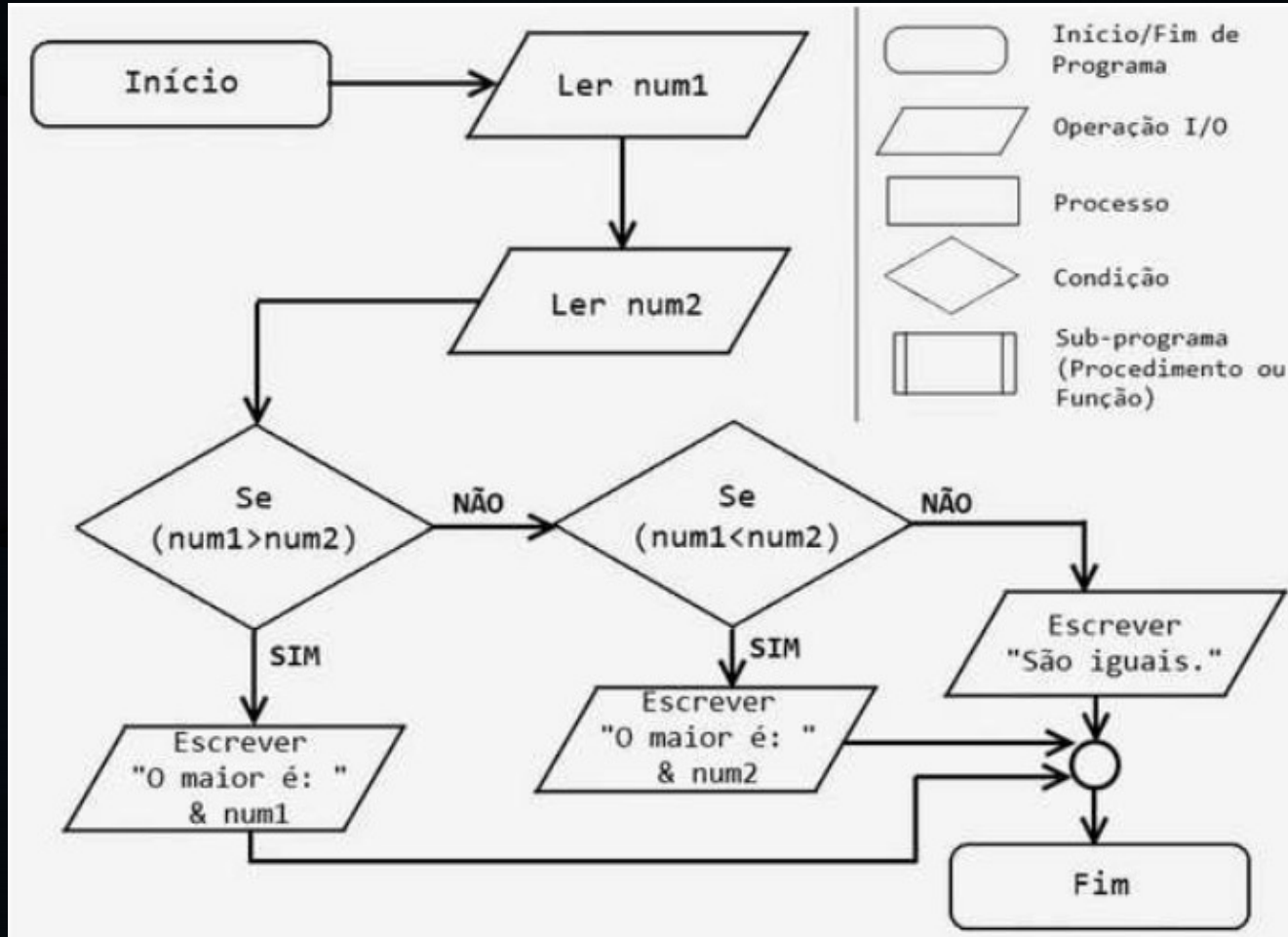
Faça um shell script semelhante ao exercício anterior porém sem utilizar o comando “read”, de forma que o site seja passado por parâmetro:

- Em seguida faça um ping para o site e salve o seu conteúdo no arquivo ping.txt (nenhum dado pode ser exibido na tela), incremente as saídas neste arquivo
- Por fim, deverá ser exibido na tela “Você digitou X sites”.

IV – Estruturas de Controle de Fluxo



IV – Estruturas de Controle de Fluxo



IV – Estruturas de Controle de Fluxo

If, test, for, while, until, case

IV – Estruturas de Controle de Fluxo

If [condição]
then

<comando1>
<comando2>...

else

<comando3>
<comando4>...

fi

Se executado com sucesso (\$?=0)
Então

Execute
Estes comandos
Se não

Execute
Estes comandos

Fim

IV – Estruturas de Controle de Fluxo

If [condição1]
then

<comando1>

elif [condição2]

<comando2>

elif [condição3]

<comando3>

fi

Se executado com sucesso (\$?=0)
Então

Execute o comando1

Se a cond1 não satisfizer, mas sim a 2

Execute comando2

Se a cond2 não satisfizer, mas sim a 3

Execute comando3

Fim

IV – Estruturas de Controle de Fluxo

test

Uso:

`test 3 -gt 4`

Ou

`[3 -gt 4]`

| | |
|-----|---------------------------------|
| -eq | Igual |
| -ne | Diferente |
| -gt | Maior |
| -lt | Menor |
| -o | Ou |
| -d | Se for um diretório |
| -e | Se existir |
| -z | Se estiver vazio |
| -f | Se conter texto |
| -o | Se o usuário for o dono |
| -r | Se o arquivo pode ser lido |
| -w | Se o arquivo pode ser alterado |
| -x | Se o arquivo pode ser executado |

IV – Estruturas de Controle de Fluxo

If [4 -gt 3] # Se 4 for maior que 3

If [\$2 -lt 4] # Se o segundo parâmetro for menor que 4

If [\$var1 -eq \$var2] # Se a variável 1 for igual a variável 2

If [`ls /teste` -z] # Se diretório estiver vazio

If [bola = bola] # Comparação entre strings

If [\$var3 = bola] # Compara o conteúdo da variável 3 com a string “bola”

IV – Estruturas de Controle de Fluxo



Laços ou Loop em Shell Script

IV – Estruturas de Controle de Fluxo

```
for ((x=0;x<=13;x++))
```

```
do
```

```
    echo "$x"
```

```
done
```

Cria um laço para contar de 0 a 13

Faça

Exiba o valor da variável

Feito

IV – Estruturas de Controle de Fluxo

for i in \$(seq 10) # i assumirá valores de 0 a 10

for i in \$(cat arquivo.txt) # i assumirá o valor de cada
linha do arquivo

for i in param[1-5] # i assumirá valores dos parâmetros 1 a 5

for ((x=0;x<=13;x++)) # os comandos do laço assumirão os
valores de 0 a 13

IV – Estruturas de Controle de Fluxo

```
while [expressão]  
do  
    <comando1>  
done
```

```
Enquanto $?=0 (bem sucedido)  
Faça  
    Execute o comando1  
Feito
```


IV – Estruturas de Controle de Fluxo

until [expressão]
do
 <comando1>
done

Enquanto $\$?=1$ (sem sucesso)
Faça
 Execute o comando1
Feito

IV – Estruturas de Controle de Fluxo

case \$var in

1)

<comando1>

2)

<comando2>

*)

<comando3>

esac

No caso da variável var receber os valores:

[\$var = 1]

Execute o comando1

[\$var = 2]

Execute o comando1

[\$var -ne 1] || [\$var -ne 2]

Execute o comando3

Fim

Exercício 4

Faça um shell script que utilize um laço para enviar pacotes icmp echo request para os endereços de IP do 192.168.100.1 ao 192.168.100.10, suprimindo as saídas padrão e de erro.

- Se o host estiver ativo deverá aparecer “Host X → ativo”
- Caso contrário deverá aparecer “Host X → inativo”

V – Tratamento de Conteúdo



V – Tratamento de Conteúdo

grep, cut, tr, sed, awk

V – Tratamento de Conteúdo

grep



grep – regex sim ou não

egrep – regex completas

fgrep – regex s/ metacaracteres

V – Tratamento de Conteúdo

```
user@kali:~# grep [palavra-chave] [arquivo]
```

V – Tratamento de Conteúdo

cut

-d – delimitador

-f – coluna

-c – conta caracteres

V – Tratamento de Conteúdo

```
user@kali:~# grep 'root' /etc/passwd | tr ":" "~"
```

```
user@kali:~# root~x~0~0~root~/root~/bin/bash
```

V – Tratamento de Conteúdo

```
user@kali:~# sed [endereço] [ação] [arquivo]
```

V – Tratamento de Conteúdo

sed

| AÇÃO |
|---|
| d - deletar |
| p – imprime mais uma vez a linha ou palavra |
| b – não faça nada nestas linhas |
| q – ao encontrar algo, saia. |
| s – substitui uma palavra por outra |
| -n – silencia o sed, mostrando apenas a linha que casa. |
| ! - utilizado como negação |


VI – Expressões Regulares




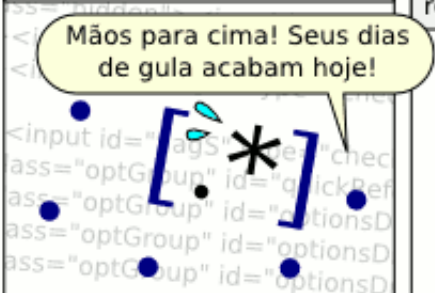




VI – Expressões Regulares

NERDSON
NÃO VAI À ESCOLA

helloworld() ;

 creative commons
BY-NC-SA
nerdson.com

| | | |
|--|--|--|
| <p>Aquele metacaractere era rico, muito rico. Mas era ganancioso demais, e não media esforços para conseguir o que queria.</p> | <p>Sua ambição não tinha limites. Isso o levou à sua própria destruição. Foi acusado de sonegar impostos, trabalho escravo infantil e tráfico de drogas.</p> | <p>Acabou entrando para a lista dos criminosos mais procurados do mundo.</p> |
|  |  PROCURA-SE |  |
| <p>Sem poderes, foi capturado durante uma tentativa falha de parsing num código html.</p> | <p>Depois de passar 12 anos na prisão e frequentar vários grupos de apoio, ele estava recuperado.</p> | <p>E dentro da lista, todo mundo é normal.</p> |
| <p>Mãos para cima! Seus dias de gula acabam hoje!</p>  |  |  |

VI – Expressões Regulares

Representantes

| meta | mnemônico | função |
|--------|--------------|--------------------------------|
| ----- | | |
| . | ponto | um caractere qualquer |
| [...] | lista | lista de caracteres permitidos |
| [^...] | lista negada | lista de caracteres proibidos |

VI – Expressões Regulares

Quantificadores

| meta | mnemônico | função |
|-------|-----------|------------------|
| ----- | | |
| ? | opcional | zero ou um |
| * | asterisco | zero, um ou mais |
| + | mais | um ou mais |
| {n,m} | chaves | de n até m |

VI – Expressões Regulares

Âncoras

| meta | mnemônico | função |
|-------|-------------|--------------------------|
| ----- | | |
| ^ | circunflexo | início da linha |
| \$ | cifrão | fim da linha |
| \b | borda | início ou fim de palavra |

VI – Expressões Regulares

Outros

| meta | mnemônico | função |
|---------|------------|-------------------------------|
| ----- | | |
| \c | escape | torna literal o caractere c |
| | ou | ou um ou outro |
| (...) | grupo | delimita um grupo |
| \1...\9 | retrovisor | texto casado nos grupos 1...9 |

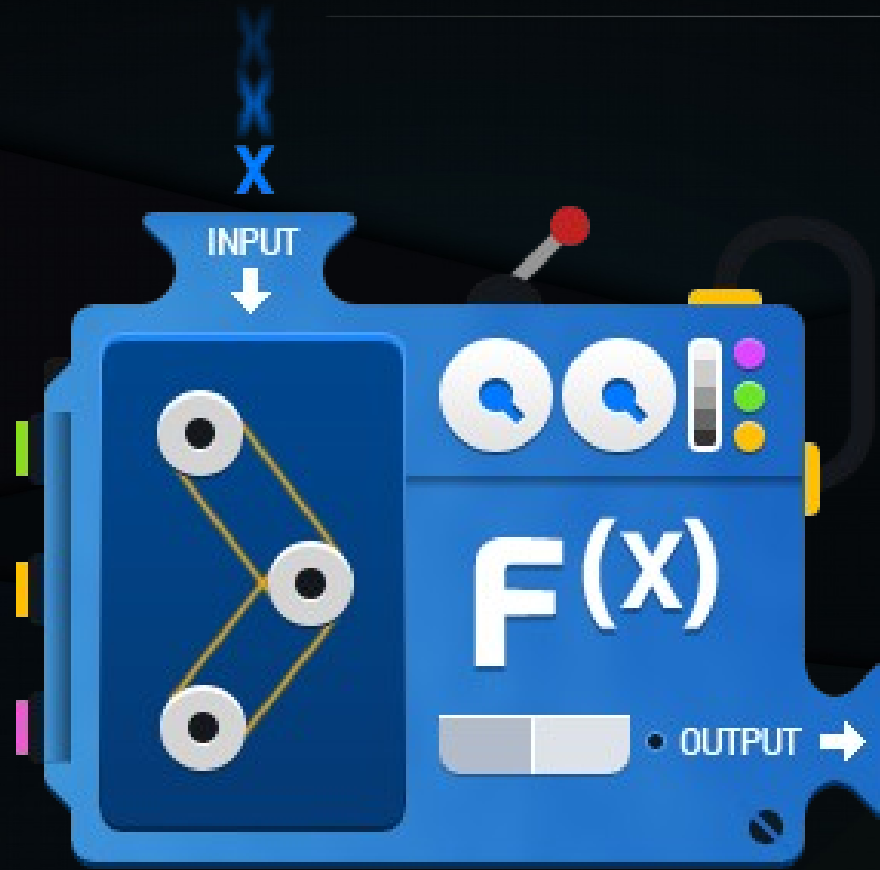
Exercício 5

De posse do arquivo Exp_reg.txt faça um shell script que manipule o arquivo para que ele liste apenas os protocolos, seguidos pelas portas e a especificação, como no exemplo a seguir:

ftp (21) transferência de arquivos
ssh (22) protocolo de acesso remoto

Obs: as portas podem ir de 0 até 65535.

VII – Funções



VII – Funções

“ Função é um bloco de comandos que, após ser declarada, pode ser invocada pelo programador de maneira similar aos comandos nativos do Shell...”

VII – Funções

#!/bin/bash

mostrar () {

Declaração de função

ls -l

Bloco de comandos

sleep 4

clear

}

mostrar

Execução da função

VII – Funções

`#!/bin/bash`

`source /root/script`

`# Invocando a função de outro arquivo`

Exercício 6

Crie um programa que possua 3 funções:

- Descoberta de IP do alvo
- Descoberta do CNPJ do alvo
- Listar os redirecionamentos externos do alvo

Obs1: O usuário deverá escolher durante a execução do programa qual a opção ele deseja executar no momento.

Obs2: O programa deverá estar sempre em execução na tela após o término de execução da ação desejada pelo usuário.

Boas Práticas em Shell Script



Boas Práticas em Shell Script

- ✓ Comentários e descrição do seu script são fundamentais
- ✓ Declare suas variáveis logo no início
- ✓ Identação, please!
- ✓ Use nomes coerentes. Evite “\$var”, “funcao ()”, “teste.sh”
- ✓ Valide seus comandos antes de iniciar um script
- ✓ Regex demais atrapalha, cuidado...
- ✓ Outros podem usar seu script, pense nisso ;)



BASH
THE BOURNE-AGAIN SHELL

Shell Script

Vinícius Vieira

vinicius@sejalivre.org