

# Introdução ao Linux

## Aula VII: Shell Script

*Prof. Dr. Marcelo Bianchi*

*Israel Dragone, Jamison Assunção  
Leonardo Fabricius e Rafael Monteiro*

26 de julho de 2018



INSTITUTO DE ASTRONOMIA,  
GEOFÍSICA E CIÊNCIAS  
ATMOSFÉRICAS

# Relembrando...

## O que é shell??

- ▶ O shell é uma interface (programa) que interpreta comandos dados pelo usuário para o sistema operacional.

# Relembrando...

## O que é shell??

- ▶ O shell é uma interface (programa) que interpreta comandos dados pelo usuário para o sistema operacional.

## Existem vários shell's

- ▶ Bourn Shell (sh);
- ▶ C Shell (csh);
- ▶ Korn Shell (ksh): mais popular no Unix;
- ▶ Bourn Again Shell (bash): mais popular no Linux, é o shell padrão do Debian GNU/Linux.

# Programação em Shell

## O que é Shell Script?

O Shell Script inicialmente foi desenvolvido para ser o interpretador de linha de comando do UNIX. Sendo basicamente, um arquivo (script) contendo uma sequência de comandos que vão sendo interpretados pelo shell na ordem em que aparecem.

Os mesmos comandos dados no terminal são aceitos dentro do script exatamente com a mesma sintaxe (cd, ls, pwd, mkdir, echo etc).

# Programação em Shell

## O que é Shell Script?

O Shell Script inicialmente foi desenvolvido para ser o interpretador de linha de comando do UNIX. Sendo basicamente, um arquivo (script) contendo uma sequência de comandos que vão sendo interpretados pelo shell na ordem em que aparecem.

Os mesmos comandos dados no terminal são aceitos dentro do script exatamente com a mesma sintaxe (cd, ls, pwd, mkdir, echo etc).

## Variáveis de ambiente no Linux - “printenv”

# Programação em Shell

## O que é Shell Script?

O Shell Script inicialmente foi desenvolvido para ser o interpretador de linha de comando do UNIX. Sendo basicamente, um arquivo (script) contendo uma sequência de comandos que vão sendo interpretados pelo shell na ordem em que aparecem.

Os mesmos comandos dados no terminal são aceitos dentro do script exatamente com a mesma sintaxe (cd, ls, pwd, mkdir, echo etc).

## Variáveis de ambiente no Linux - “printenv”

- ▶ PATH → Armazena o caminho para os diretórios que contém os comandos do Linux;

# Programação em Shell

## O que é Shell Script?

O Shell Script inicialmente foi desenvolvido para ser o interpretador de linha de comando do UNIX. Sendo basicamente, um arquivo (script) contendo uma sequência de comandos que vão sendo interpretados pelo shell na ordem em que aparecem.

Os mesmos comandos dados no terminal são aceitos dentro do script exatamente com a mesma sintaxe (cd, ls, pwd, mkdir, echo etc).

## Variáveis de ambiente no Linux - “printenv”

- ▶ PATH → Armazena o caminho para os diretórios que contém os comandos do Linux;
- ▶ TERM → Define o terminal padrão;

# Programação em Shell

## O que é Shell Script?

O Shell Script inicialmente foi desenvolvido para ser o interpretador de linha de comando do UNIX. Sendo basicamente, um arquivo (script) contendo uma sequência de comandos que vão sendo interpretados pelo shell na ordem em que aparecem.

Os mesmos comandos dados no terminal são aceitos dentro do script exatamente com a mesma sintaxe (cd, ls, pwd, mkdir, echo etc).

## Variáveis de ambiente no Linux - “printenv”

- ▶ PATH → Armazena o caminho para os diretórios que contém os comandos do Linux;
- ▶ TERM → Define o terminal padrão;
- ▶ HOME → Indica o diretório pessoal do usuário em questão;



# Programação em Shell

## O que é Shell Script?

O Shell Script inicialmente foi desenvolvido para ser o interpretador de linha de comando do UNIX. Sendo basicamente, um arquivo (script) contendo uma sequência de comandos que vão sendo interpretados pelo shell na ordem em que aparecem.

Os mesmos comandos dados no terminal são aceitos dentro do script exatamente com a mesma sintaxe (cd, ls, pwd, mkdir, echo etc).

## Variáveis de ambiente no Linux - “printenv”

- ▶ PATH → Armazena o caminho para os diretórios que contém os comandos do Linux;
- ▶ TERM → Define o terminal padrão;
- ▶ HOME → Indica o diretório pessoal do usuário em questão;
- ▶ USER → Guarda o nome do usuário no momento;

# Programação em Shell

## O que é Shell Script?

O Shell Script inicialmente foi desenvolvido para ser o interpretador de linha de comando do UNIX. Sendo basicamente, um arquivo (script) contendo uma sequência de comandos que vão sendo interpretados pelo shell na ordem em que aparecem.

Os mesmos comandos dados no terminal são aceitos dentro do script exatamente com a mesma sintaxe (cd, ls, pwd, mkdir, echo etc).

## Variáveis de ambiente no Linux - “printenv”

- ▶ PATH → Armazena o caminho para os diretórios que contém os comandos do Linux;
- ▶ TERM → Define o terminal padrão;
- ▶ HOME → Indica o diretório pessoal do usuário em questão;
- ▶ USER → Guarda o nome do usuário no momento;
- ▶ SHELL → Guarda o valor do shell padrão.

# Script's Padrões

~/.bashrc

# Script's Padrões

~/.bashrc

- ▶ O arquivo oculto “.bashrc” é um script;

# Script's Padrões

~/.bashrc

- ▶ O arquivo oculto “.bashrc” é um script;
- ▶ É executado sempre que uma nova sessão de terminal é iniciada;

# Script's Padrões

## ~/.bashrc

- ▶ O arquivo oculto “.bashrc” é um script;
- ▶ É executado sempre que uma nova sessão de terminal é iniciada;
- ▶ São shell's próprios do usuário não sendo necessário um login.

# Script's Padrões

## ~/.bashrc

- ▶ O arquivo oculto “.bashrc” é um script;
- ▶ É executado sempre que uma nova sessão de terminal é iniciada;
- ▶ São shell's próprios do usuário não sendo necessário um login.

## ~/.bash\_profile ou ~/.profile

# Script's Padrões

## ~/.bashrc

- ▶ O arquivo oculto “.bashrc” é um script;
- ▶ É executado sempre que uma nova sessão de terminal é iniciada;
- ▶ São shell's próprios do usuário não sendo necessário um login.

## ~/.bash\_profile ou ~/.profile

- ▶ O arquivo oculto “.profile” também é um script;



# Script's Padrões

## ~/.bashrc

- ▶ O arquivo oculto “.bashrc” é um script;
- ▶ É executado sempre que uma nova sessão de terminal é iniciada;
- ▶ São shell's próprios do usuário não sendo necessário um login.

## ~/.bash\_profile ou ~/.profile

- ▶ O arquivo oculto “.profile” também é um script;
- ▶ É executado sempre que for feito login em um sistema remoto (SSH);

# Script's Padrões

## ~/.bashrc

- ▶ O arquivo oculto “.bashrc” é um script;
- ▶ É executado sempre que uma nova sessão de terminal é iniciada;
- ▶ São shell's próprios do usuário não sendo necessário um login.

## ~/.bash\_profile ou ~/.profile

- ▶ O arquivo oculto “.profile” também é um script;
- ▶ É executado sempre que for feito login em um sistema remoto (SSH);
- ▶ Este script subscreve as configurações dadas, anteriormente, pelo “.bashrc”.

# Configurando .bashrc ou .profile

## Adicionando diretório no PATH

Você pode tornar seus scripts acessíveis de qualquer local da máquina, sem precisar especificar o caminho completo. Basta adicionar no PATH.

# Configurando .bashrc ou .profile

## Adicionando diretório no PATH

Você pode tornar seus scripts acessíveis de qualquer local da máquina, sem precisar especificar o caminho completo. Basta adicionar no PATH.

```
$ export PATH=$PATH:$HOME/caminho_para_o_diretório
```

# Configurando .bashrc ou .profile

## Adicionando diretório no PATH

Você pode tornar seus scripts acessíveis de qualquer local da máquina, sem precisar especificar o caminho completo. Basta adicionar no PATH.

```
$ export PATH=$PATH:$HOME/caminho_para_o_diretório
```

## Criando um apelido para o programa

O comando “alias” permite criar apelidos para comandos no sistema. Assim, podemos renomear o programa para não precisar mais digitar seu nome completo.

# Configurando .bashrc ou .profile

## Adicionando diretório no PATH

Você pode tornar seus scripts acessíveis de qualquer local da máquina, sem precisar especificar o caminho completo. Basta adicionar no PATH.

```
$ export PATH=$PATH:$HOME/caminho_para_o_diretório
```

## Criando um apelido para o programa

O comando “alias” permite criar apelidos para comandos no sistema. Assim, podemos renomear o programa para não precisar mais digitar seu nome completo.

```
$ alias ll='ls -l'
```

## Configurando .bashrc ou .profile

Ao fechar o terminal todas as alterações feitas pelos comandos acima serão apagados.

Para torna-las permanentes é necessário adicionar o código no script `~/.bashrc`

## Configurando .bashrc ou .profile

Ao fechar o terminal todas as alterações feitas pelos comandos acima serão apagados.

Para torna-las permanentes é necessário adicionar o código no script `~/.bashrc`

Contudo `~/.bashrc` apenas é executado quando se abre um novo terminal, nos forçando a tarefa morosa de abrir e fechar novos terminais continuamente.



## Configurando .bashrc ou .profile

Ao fechar o terminal todas as alterações feitas pelos comandos acima serão apagados.

Para torna-las permanentes é necessário adicionar o código no script `~/.bashrc`

Contudo `~/.bashrc` apenas é executado quando se abre um novo terminal, nos forçando a tarefa morosa de abrir e fechar novos terminais continuamente.

**source** arquivo [argumentos]

Este comando pega o conteúdo do arquivo especificado e o transmite para o interpretador(bash) como um script de texto.

## Configurando .bashrc ou .profile

Ao fechar o terminal todas as alterações feitas pelos comandos acima serão apagados.

Para torna-las permanentes é necessário adicionar o código no script `~/.bashrc`

Contudo `~/.bashrc` apenas é executado quando se abre um novo terminal, nos forçando a tarefa morosa de abrir e fechar novos terminais continuamente.

### **source** arquivo [argumentos]

Este comando pega o conteúdo do arquivo especificado e o transmite para o interpretador(bash) como um script de texto.

`$ source .bashrc` → Atualiza as configurações do terminal.

# Exercício - Primeiro Bloco

1. Use o comando “printenv” e descubra o conteúdo das variáveis PATH, HOME, SHELL e USER para o .bashrc;
2. Faça a mesma coisa que o exercício acima, porém para o .profile;  
**SUGESTÃO:** Use grep e ssh quando necessário;
3. Adicione “apelidos” permanentes para os comandos que você mais gostou;
4. Adicione um caminho qualquer dentro do PATH, não necessariamente precisa ter um comando relacionado;
5. Mude o nome de usuário (USER);  
**NÃO** coloque isso no .bashrc

# Atribuição de variáveis

Em shell não é necessário declarar o tipo nem o tamanho da variável, basta fazer a atribuição.

# Atribuição de variáveis

Em shell não é necessário declarar o tipo nem o tamanho da variável, basta fazer a atribuição.

## Exemplo

# Atribuição de variáveis

Em shell não é necessário declarar o tipo nem o tamanho da variável, basta fazer a atribuição.

## Exemplo

► `var1 = 1`

# Atribuição de variáveis

Em shell não é necessário declarar o tipo nem o tamanho da variável, basta fazer a atribuição.

## Exemplo

- ▶ `var1 = 1`
- ▶ `var2 = 2.5`

# Atribuição de variáveis

Em shell não é necessário declarar o tipo nem o tamanho da variável, basta fazer a atribuição.

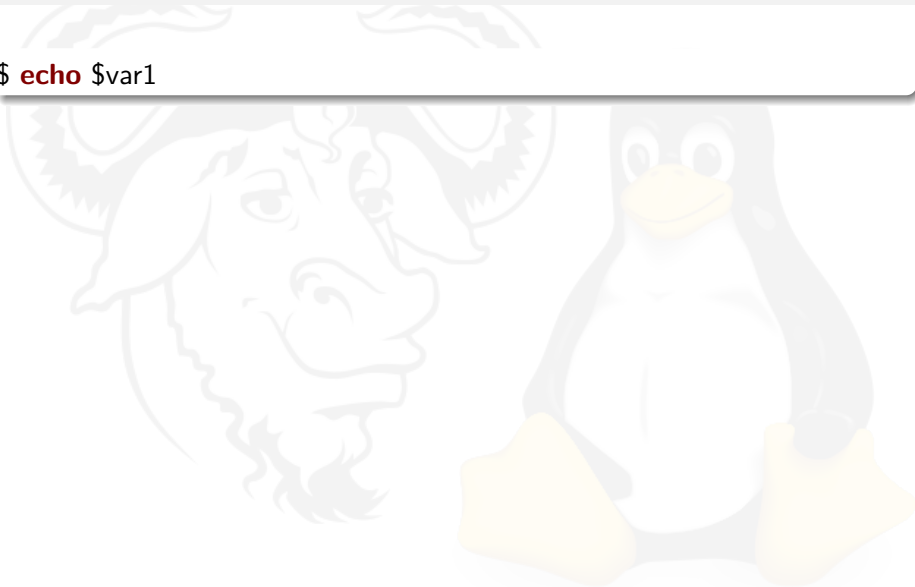
## Exemplo

- ▶ `var1 = 1`
- ▶ `var2 = 2.5`
- ▶ `var3 = "hello world"`



echo → Imprimindo na tela

```
$ echo $var1
```



echo → Imprimindo na tela

```
$ echo $var1
```

```
1
```

## echo → Imprimindo na tela

```
$ echo $var1
```

```
1
```

```
$ echo "$var3, var1 = $var1 e var2 = $var2"
```

## echo → Imprimindo na tela

```
$ echo $var1
```

```
1
```

```
$ echo "$var3, var1 = $var1 e var2 = $var2"
```

```
hello world, var1 = 1 e var2 = 2.5  
(aspas duplas reconhecem $, \ e `)
```

## echo → Imprimindo na tela

```
$ echo $var1
```

```
1
```

```
$ echo "$var3, var1 = $var1 e var2 = $var2"
```

```
hello world, var1 = 1 e var2 = 2.5  
(aspas duplas reconhecem $, \ e `)
```

```
$ echo '$var3, var1 = $var1 e var2 = $var2'
```

## echo → Imprimindo na tela

```
$ echo $var1
```

```
1
```

```
$ echo "$var3, var1 = $var1 e var2 = $var2"
```

```
hello world, var1 = 1 e var2 = 2.5
```

(aspas duplas reconhecem \$, \ e `)

```
$ echo '$var3, var1 = $var1 e var2 = $var2'
```

```
$var3, var1 = $var1 e var2 = $var2
```

(aspas simples, protegem completamente a string)

echo → Imprimindo na tela

Opções Importantes

## echo → Imprimindo na tela

### Opções Importantes

- ▶ **echo -n** → Não pula linha automaticamente ao final do echo;



# echo → Imprimindo na tela

## Opções Importantes

- ▶ **echo -n** → Não pula linha automaticamente ao final do echo;
- ▶ **echo -e** → Reconhece caracteres de escape como \;

# echo → Imprimindo na tela

## Opções Importantes

- ▶ **echo -n** → Não pula linha automaticamente ao final do echo;
- ▶ **echo -e** → Reconhece caracteres de escape como \;
  - ▶ **\n** → pula linha;

# echo → Imprimindo na tela

## Opções Importantes

- ▶ **echo -n** → Não pula linha automaticamente ao final do echo;
- ▶ **echo -e** → Reconhece caracteres de escape como \;
  - ▶ **\n** → pula linha;
  - ▶ **\t** → tabulação horizontal;

## Exemplo

```
$ echo -e "Alunos do 2º ano: \n André \t Beatriz \v Camila"
```

# echo → Imprimindo na tela

## Opções Importantes

- ▶ **echo -n** → Não pula linha automaticamente ao final do echo;
- ▶ **echo -e** → Reconhece caracteres de escape como \;
  - ▶ **\n** → pula linha;
  - ▶ **\t** → tabulação horizontal;
  - ▶ **\v** → tabulação vertical;

## Exemplo

\$ **echo -e** "Alunos do 2º ano: \n André \t Beatriz \v Camila"

Alunos do 2º ano:

André

Beatriz

Camila

# read → Leitura de uma interação do usuário

## Leitura Simples

\$ **read** Nome ; *Usuário = Peter*

\$ **echo** \$Nome

# read → Leitura de uma interação do usuário

## Leitura Simples

\$ **read** Nome ; *Usuário = Peter*

\$ **echo** \$Nome

Peter

# read → Leitura de uma interação do usuário

## Leitura Simples

```
$ read Nome ; Usuário = Peter
```

```
$ echo $Nome
```

```
Peter
```

## Leitura Composta

```
$ read v1 v2 v3 ; Usuário = 10 Oi !
```

```
$ echo "$v1 $v2 $v3"
```

# read → Leitura de uma interação do usuário

## Leitura Simples

```
$ read Nome ; Usuário = Peter
```

```
$ echo $Nome
```

```
Peter
```

## Leitura Composta

```
$ read v1 v2 v3 ; Usuário = 10 Oi !
```

```
$ echo "$v1 $v2 $v3"
```

```
10 Oi !
```



# read → Leitura de uma interação do usuário

## Leitura Simples

```
$ read Nome ; Usuário = Peter
```

```
$ echo $Nome
```

```
Peter
```

## Leitura Composta

```
$ read v1 v2 v3 ; Usuário = 10 Oi !
```

```
$ echo "$v1 $v2 $v3"
```

```
10 Oi !
```

## Leitura com limite de caracteres (não é necessário pressionar enter)

```
$ read -n4 ano ; Usuário = 2018
```

```
$ echo $ano
```

# read → Leitura de uma interação do usuário

## Leitura Simples

```
$ read Nome ; Usuário = Peter
```

```
$ echo $Nome
```

```
Peter
```

## Leitura Composta

```
$ read v1 v2 v3 ; Usuário = 10 Oi !
```

```
$ echo "$v1 $v2 $v3"
```

```
10 Oi !
```

## Leitura com limite de caracteres (não é necessário pressionar enter)

```
$ read -n4 ano ; Usuário = 2018
```

```
$ echo $ano
```

```
2018
```

# read → Leitura de uma interação do usuário

## Leitura na mesma linha

```
$ read -p "Cor dos olhos: " cor ; Usuário = azul
```

```
$ echo $cor
```

# read → Leitura de uma interação do usuário

## Leitura na mesma linha

```
$ read -p "Cor dos olhos: " cor ; Usuário = azul
```

```
$ echo $cor
```

```
azul
```

## read → Leitura de uma interação do usuário

### Leitura na mesma linha

```
$ read -p "Cor dos olhos: " cor ; Usuário = azul
```

```
$ echo $cor
```

```
azul
```

### Leitura sem exibir na tela

```
$ read -s senha;
```

```
$ echo $senha
```

## read → Leitura de uma interação do usuário

### Leitura na mesma linha

```
$ read -p "Cor dos olhos: " cor ; Usuário = azul
```

```
$ echo $cor
```

```
azul
```

### Leitura sem exibir na tela

```
$ read -s senha;
```

```
$ echo $senha
```

```
????
```

## read → Leitura de uma interação do usuário

### Leitura na mesma linha

```
$ read -p "Cor dos olhos: " cor ; Usuário = azul
```

```
$ echo $cor
```

```
azul
```

### Leitura sem exibir na tela

```
$ read -s senha;
```

```
$ echo $senha
```

```
????
```

### Leitura com pausa em carácter especificado

```
$ read -d'a' nome; Usuário = Leonardo
```

```
$ echo $nome
```

## read → Leitura de uma interação do usuário

### Leitura na mesma linha

```
$ read -p "Cor dos olhos: " cor ; Usuário = azul
```

```
$ echo $cor
```

```
azul
```

### Leitura sem exibir na tela

```
$ read -s senha;
```

```
$ echo $senha
```

```
????
```

### Leitura com pausa em carácter especificado

```
$ read -d'a' nome; Usuário = Leonardo
```

```
$ echo $nome
```

```
Leon
```



## Exercício - Segundo Bloco

Faça um arquivo que efetue um cadastro de uma pessoa. O formulario deve conter **Nome, Idade, Nacionalidade, E-mail e Palavra-Chave**.

No final do *script* mostre os dados cadastrados.

### Dicas

Para criar um arquivo execute:

```
$ gedit nome_do_arquivo
```

Para processar o arquivo execute:

```
$ bash nome_do_arquivo
```

### Exemplo - Não vale copiar.

```
$ echo "Nome: "
```

```
$ read nome
```

```
$ echo "Idade: "
```

```
$ read idade
```

...