

Notas de Aula gentilmente disponibilizadas pelo aluno Pedro Henrique

Aula 9 - 103.1 - Variáveis de Ambiente

O que veremos nesta aula:

Breve Introdução sobre variáveis de ambiente	2
Variáveis.....	3
Como declarar uma variável	3
Variáveis Locais e Globais	3
Comando set.....	5
Comando env	6
Diferenças os comandos entre set e env	6
Como remover variáveis criadas.....	8
O comando unset.....	8
Variáveis Pré-definidas do ambiente Linux	8

Breve Introdução sobre variáveis de ambiente

Variável de ambiente é uma variável de um sistema operacional que geralmente contém informações sobre o sistema, caminhos de diretórios específicos no sistema de arquivos e as preferências do utilizador. Ela pode afetar a forma como um processo se comporta, e cada processo pode ler e escrever variáveis de ambiente.

Em todos os sistemas Unix, cada processo possui seu conjunto privado de variáveis de ambiente. **Por padrão, quando um processo é criado ele herda uma cópia das variáveis que foram exportadas no ambiente do processo pai.** *Todos os tipos de Unix assim como o DOS e o Microsoft Windows possuem variáveis de ambiente; entretanto, variáveis para funções parecidas entre os sistemas possuem nomes distintos.* Programas podem acessar os valores das variáveis de ambiente para efeitos de configuração.

Shell scripts e arquivos de lote usam variáveis para armazenar dados temporários e também para comunicar dados e preferências a processos filhos.

No Unix, as variáveis de ambiente são normalmente inicializadas durante a inicialização do sistema, e no início de cada sessão. Esse assunto será estudado melhor no tópico 5.

As variáveis podem ser usadas tanto por scripts quanto pela linha de comando. São geralmente referenciadas usando-se símbolos especiais na frente ou nas extremidades no nome da variável. Por exemplo, Unix usa-se o \$. O **cifrão**, quando iniciamos o comando identifica que é uma variável de ambiente, se não colocarmos o cifrão ele apenas imprime na tela o conteúdo, como vimos na aula anterior:

Se dermos o comando abaixo, ele nos retornará o caminho de cara path de variável no sistema:

```
$ echo $PATH
```

```
lpil@linux:~/Desktop$ echo $PATH
/home/lpil/bin:/home/lpil/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
lpil@linux:~/Desktop$
```

Sem o cifrão ele apenas imprime a palavra na tela:

```
$ echo PATH
```

```
lpil@linux:~/Desktop$ echo PATH
PATH
lpil@linux:~/Desktop$
```

Variáveis

Como declarar uma variável

Para declarar(criar) uma variável no ambiente Linux seguimos o princípio de qualquer linguagem de programação, **primeiros damos o nome e depois o valor**. Exemplo no terminal:

```
NOME_VARIAVEL=valor
```

Acima dizemos que o nome da variável é **NOME_VARIAVEL** e depois declaramos o seu valor, ou seja, **NOME_VARIAVEL** é o mesmo que *valor*.

```
lpil@linux:~/Desktop$ NOME_VARIAVEL=valor  
lpil@linux:~/Desktop$ echo $NOME_VARIAVEL  
valor  
lpil@linux:~/Desktop$
```

Para ficar mais claro, vamos declarar mais uma variável:

```
CURSOLINUX=lpil
```

Se digitarmos no terminal o comando abaixo ele nos mostrar o valor de atribuímos a variável **CURSOLINUX** que é *lpil*:

```
echo $CURSOLINUX
```

```
lpil@linux:~/Desktop$ CURSOLINUX=lpil  
lpil@linux:~/Desktop$ echo $CURSOLINUX  
lpil  
lpil@linux:~/Desktop$
```

Note que, primeiro declaramos a variável e logo depois consultamos qual o valor dela.

Variáveis Locais e Globais

Importante frisar que, quando declaramos uma variável da forma que é explicado acima, ele fica disponível e visível somente em âmbito local, ou seja, a variável só é visível na seção do shell que está aberta, quaisquer seções abertas posteriormente não encontrarão as variáveis criadas.

Vejamos, como já criamos as variáveis **NOME_VARIAVEL** e **CURSOLINUX** vamos iniciar um novo bash dentro do próprio bash que está aberto, lembrando que estas variáveis ainda estão em nível local:

```
bash
```

```
lpil@linux:~$ bash
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

lpil@linux:~$
```

Note que estamos em outra seção, agora faça o teste e tente consultar o valor das variáveis `NOME_VARIAVEL` e `CURSOLINUX`:

<code>echo \$NOME_VARIAVEL</code>
<code>echo \$CURSOLINUX</code>

Não nos voltará nenhum valor, pois estas variáveis estão armazenadas apenas localmente para aquele usuário e aquela seção anterior. Para voltar ao **bash** anterior digite “**exit**”. Isso vale também caso o **bash** seja fechado, ou seja, se fecharmos o terminal as variáveis também serão apagadas.

Comando export

Para que os processos filhos consigam ver essas variáveis (global) usamos o comando **export** (export `NOME_VARIAVEL`), pois só após exportarmos essas variáveis é que os processos filhos poderão enxergar essas variáveis, seguindo a hierarquia. Toda seção que for aberta durante o bash atual, será “filha” dela, assim herdando as variáveis. **O comando export só funciona para processos que são originados a partir do bash atual.** Cuidado para não confundir com um novo terminal, pois é totalmente independente.

Vamos declarar a variável `TESTE=Linux`:

<code>TESTE=Linux</code>

```
lpil@linux:~/Desktop$ TESTE=Linux
lpil@linux:~/Desktop$
```

Agora, vamos usar o arquivo da aula para testar a variável, o arquivo se chama `Script_Variavel.sh`, vamos ver o que tem dentro dele:

<code>cat Script_Variavel.sh</code>

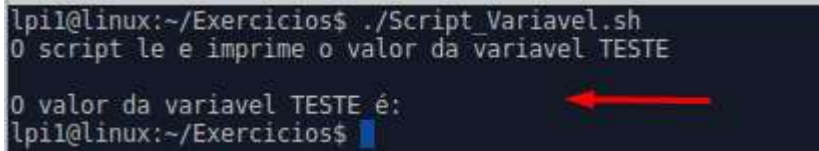
```
lpil@linux:~/Exercicios$ cat Script_Variavel.sh
#!/bin/bash
echo "O script le e imprime o valor da variavel TESTE"
echo " "
echo "O valor da variavel TESTE é:" $TESTE

lpil@linux:~/Exercicios$
```

Podemos ver que durante o script há campo chamando o valor da variável TESTE, vamos lembrar que quando tem o **cifrão “\$”** quer dizer que é uma variável.

Se fizermos o comando, vamos ver que ficará um campo sem o valor da variável.

```
./Script_Variavel.sh
```

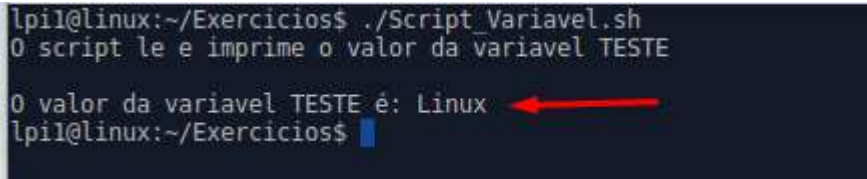


```
lpil@linux:~/Exercicios$ ./Script_Variavel.sh
0 script le e imprime o valor da variavel TESTE
0 valor da variavel TESTE é:
lpil@linux:~/Exercicios$
```

Isso acontece porque a variável está local, então somente o bash atual conseguiu encontra-lo. Para que toda seção a partir da atual passe a enxergar a variável TESTE temos que fazer um export.

```
export TESTE
```

Agora, se rodarmos novamente o script do exercício temos o seguinte resultado:



```
lpil@linux:~/Exercicios$ ./Script_Variavel.sh
0 script le e imprime o valor da variavel TESTE
0 valor da variavel TESTE é: Linux
lpil@linux:~/Exercicios$
```

Podemos criar e depois fazer o export nas variáveis, ou podemos criar e exportar diretamente uma variável com o comando abaixo.

```
export NOME_VARIAVEL
```

Como ver essas variáveis

Comando set

O comando **set** irá mostrar todas as variáveis, as local e as globais declaradas no bash atual.

```
set | less
```

Ainda não estudamos o comando parâmetro **less**, virá nas próximas aulas. Mas ele é usado para minimizar a quantidade de informações na tela. Após o comando, veremos várias variáveis de ambiente inclusive que declaramos.

Comando env

O comando **env** irá mostrar apenas as variáveis globais.

Diferenças os comandos entre set e env

A grande diferença entra o **set** e o **env** é que o **set** é um comando de origem do **bash**, já o **env** é um comando de uma aplicação externa, por isso ele enxerga somente as globais.

Vamos declarar uma variável ABD com o valor cde (ABC=cde).

```
lpil@linux:~/Desktop$ ABC=cde  
lpil@linux:~/Desktop$
```

Agora, vamos dar o comando para visualizar as variáveis locais e globais, o **set**.

```
set | less
```

De cara, já conseguimos ver a variável que declaramos:

```
ABC=cde  
BASH=/bin/bash  
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote:force_ign  
BASH_ALIASES=()  
BASH_ARGC=()  
BASH_ARGV=()  
BASH_CMDS=()  
BASH_COMPLETION_COMPAT_DIR=/etc/bash_completion.d  
BASH_LINENO=()  
BASH_SOURCE=()  
BASH_VERSIONINFO=([0]="4" [1]="3" [2]="48" [3]="1" [4]="release" [5]="x86_64-pc-linux-gnu")  
BASH_VERSION='4.3.48(1)-release'  
CLUTTER_BACKEND=x11  
CLUTTER_IM_MODULE=  
COLORTERM=xfce4-terminal  
COLUMNS=173  
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-ZCl5heZKzv  
DEFAULTS_PATH=/usr/share/gconf/xubuntu.default.path  
DESKTOP_SESSION=xubuntu  
DIRSTACK=()
```

Agora, vamos fazer a mesma coisa, porém com o comando **env**:

```
env | less
```

```

XDG_VTNR=7
LC_PAPER=pt_BR.UTF-8
LC_ADDRESS=pt_BR.UTF-8
XDG_SESSION_ID=c2
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/lpil
LC_MONETARY=pt_BR.UTF-8
CLUTTER_IM_MODULE=
QT_STYLE_OVERRIDE=gtk
SESSION=xubuntu
GPG_AGENT_INFO=/home/lpil/.gnupg/S.gpg-agent:0:1
GLADE_PIXMAP_PATH=
XDG_MENU_PREFIX=xfce-
SHELL=/bin/bash
TERM=xterm
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=62914564
LC_NUMERIC=pt_BR.UTF-8
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/30796
GNOME_KEYRING_CONTROL=
USER=lpil

```

Note que já não conseguimos ver a variável ABC, pois o comando **env** mostra somente as variáveis que são globais. Para que o comando **env** a encontre temos que fazer o **export**, veja:

```
export ABC=cde
```

```

lpil@linux:~/Desktop$ export ABC=cde
lpil@linux:~/Desktop$

```

Agora vamos repetir o comando env:

```
env | less
```

```

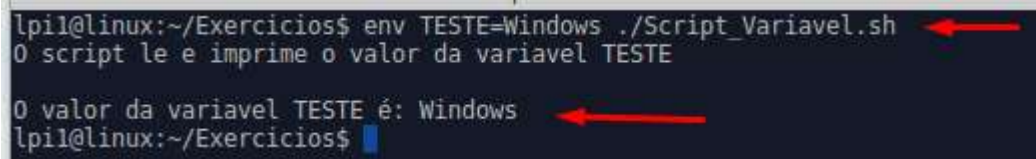
ABC=cde
XDG_VTNR=7
LC_PAPER=pt_BR.UTF-8
LC_ADDRESS=pt_BR.UTF-8
XDG_SESSION_ID=c2
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/lpil
LC_MONETARY=pt_BR.UTF-8
CLUTTER_IM_MODULE=
QT_STYLE_OVERRIDE=gtk
SESSION=xubuntu
GPG_AGENT_INFO=/home/lpil/.gnupg/S.gpg-agent:0:1
GLADE_PIXMAP_PATH=
XDG_MENU_PREFIX=xfce-
SHELL=/bin/bash
TERM=xterm
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=62914564
LC_NUMERIC=pt_BR.UTF-8
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/30796
GNOME_KEYRING_CONTROL=
USER=lpil

```

Isso acontece porque agora a variável está disponível globalmente.

Uma outra opção do **env** é que, ele pode alterar o valor uma variável de forma temporária, por exemplo, com o script do exercício, o valor da variável TESTE é Linux, vamos alterá-lo apenas na execução. (Lembrando que estamos dentro do diretório Exercícios):

```
env TESTE=Windows ./Script_Variavel
```

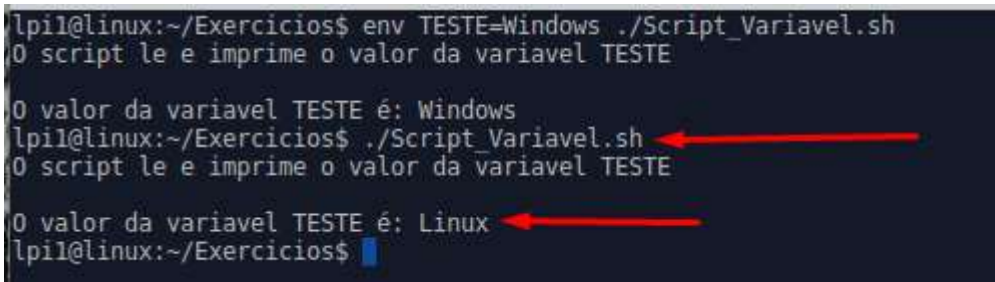


```
lpil@linux:~/Exercicios$ env TESTE=Windows ./Script_Variavel.sh
O script le e imprime o valor da variavel TESTE

O valor da variavel TESTE é: Windows
lpil@linux:~/Exercicios$
```

Agora se executarmos novamente o comando de forma normal, ele apresentará o valor verdadeiro da variável.

```
./Script_Variavel
```



```
lpil@linux:~/Exercicios$ env TESTE=Windows ./Script_Variavel.sh
O script le e imprime o valor da variavel TESTE

O valor da variavel TESTE é: Windows
lpil@linux:~/Exercicios$ ./Script_Variavel.sh
O script le e imprime o valor da variavel TESTE

O valor da variavel TESTE é: Linux
lpil@linux:~/Exercicios$
```

Como remover variáveis criadas

Da mesma forma que temos como criar uma variável, também temos como removê-las do sistema. Para isso usamos o comando **“unset”**. En

O comando unset

Sempre que quisermos excluir uma variável usamos o comando **“unset”**. Para remover o valor da variável TESTE usaremos o comando:

```
unset TESTE
```

Agora, vamos consultar a variável com o comando echo e veremos que não teremos mais o valor da variável TESTES disponível.

```
echo $TESTE
```

Variáveis Pré-definidas do ambiente Linux

Ainda sobre variáveis vale frisar que existem variáveis que já são predefinidas no sistema Linux, as quais, são carregadas durante o sistema, na aula citamos algumas que inclusive pode ser cobrado no exame.

Set | more

Aqui vão as citadas:

- HISTFILE=/home/lpi1/ .bash_history –caminho onde armazena os comandos feitos no terminal.
- HISTFILESIZE=2000 – Tamanho máximo que o arquivo terá.
- HISTSIZE=1000 – Limite máximo de linhas(comandos) no arquivo.
- HOME=/home/lpi1 – Mostra o home do usuário atual.
- LOGNAME=lpi1 – Mostra o nome do usuário que fez o login na seção atual.
- PATH= - Mostra todos os caminhos dos programas no sistema
- PWD=/home/lpi1/Exercicios – Mostra o diretório atual.
- SHELL
- TERM=xterm - Mostra qual terminal estamos usando, no caso estamos usando interface gráfica. Caso optemos por logar sem passar por interface gráfica, aparecerá TERM=tty.
- USER=lpi1 = Mostra o nome do usuário atual

Todas as variáveis podemos conferir com o comando echo.

Existem algumas variáveis de ambiente que são definidas dinamicamente pelo SHELL, é importante que conheçamos elas. Elas são identificadas pelo cifrão no início., por exemplo:

Este comando mostra o PID do processo atual.

echo \$\$

Este comando mostra o PID do último processo que executamos em background.

echo \$!

Este comando mostra o código de saída (exit code) do último do processo executado.

echo \$?

Ainda temos o "~", que contém o /home do usuário, seja ele qual for.

echo ~

Ainda podemos apontar outro usuário com o mesmo comando, ele voltará o home deste usuário:

```
echo ~root
```

Quando executamos o comando **cd ~**, vamos direto para o /home do usuário atual.

Material adicional:

<https://leonardoafonsoamorim.wordpress.com/2013/01/25/variaveis-de-ambiente-no-linux/>

<https://www.ibm.com/developerworks/br/linux/library/l-lpic1-v3-103-1/>