LXT0209-Variáveis

Post-Linkedin

Todo Shell gere um conjunto de informações de estado ao longo das sessões do Shell. Essas informações de tempo de execução podem mudar durante a sessão e influenciar o comportamento do Shell. Esses dados também são usados pelos programas para determinar aspectos da configuração do sistema. A maioria desses dados é armazenada nas chamadas *variáveis*, que serão analisadas nesse artigo.

Introdução

As variáveis são espaços de armazenamento para dados, como texto ou números. Uma vez definido, o valor de uma variável pode ser acessado posteriormente. As variáveis têm um nome, o que permite acessar uma variável específica mesmo quando seu conteúdo é alterado. Elas são uma ferramenta muito comum na maioria das linguagens de programação.

Na maioria dos Shells do Linux, existem dois tipos de variáveis:

- **Variáveis locais:** estão disponíveis apenas para o processo actual do Shell. Quando é criada uma variável local e, em seguida, iniciar outro programa nesse Shell, a variável não poderá mais ser acessada por esse programa. Como não são herdadas por sub-processos, essas variáveis são chamadas *variáveis locais*.
- **Variáveis de ambiente:** estão disponíveis tanto em uma sessão de Shell específica quanto em sub-processos gerados a partir dessa sessão de Shell. Essas variáveis podem ser usadas para transmitir dados de configuração para os comandos executados. Como os programas podem acessar essas variáveis, elas são chamadas *variáveis de ambiente*. A maioria das variáveis de ambiente aparece em letras maiúsculas (por exemplo, `PATH`, `DATE`, `USER`). Um conjunto de variáveis de ambiente padrão fornece, por exemplo, informações sobre o directório inicial ou o tipo de terminal do usuário. Em certos casos, nos referimos ao conjunto completo de todas as variáveis de ambiente como *ambiente*.

  Esses tipos de variáveis também são conhecidos como *escopo de variáveis*.

Manipulação de variáveis

Todo administrador de sistema precisa criar, modificar ou remover variáveis locais e de ambiente.

Variáveis locais

É possível configurar uma variável local usando o operador `=` (igual). Uma atribuição simples criará uma variável local:

```
$ greeting=hello
```

OBS: não se deve ter espaço antes ou depois do operador `=`.

É possível exibir qualquer variável usando o comando `echo`. O comando geralmente exibe o texto na secção de argumentos:

```
idaseg@debian:/$ greeting=hello
idaseg@debian:/$ echo greeting
greeting
```

Para aceder o valor da variável, é preciso usar `$` (cifrão) na frente do nome da variável.

```
idaseg@debian:/$ echo $greeting
hello
```

Como pode ser visto, a variável foi criada. Se abrirmos outro Shell e tente exibir o conteúdo da variável criada. Temos o seguinte resultado:

```
idaseg@debian:~$ echo $greeting

idaseg@debian:~$ _
```

Nada é exibido. Isso mostra que as variáveis existem somente em um Shell específico.

Para verificar se a variável é de fato uma variável local, podemos gerar um novo processo e verificar se esse processo consegue aceder a variável. Para isso, é preciso abrir outro Shell e executar o comando `echo`. Como o novo Shell roda em um novo processo, ele não herdará as variáveis locais do processo pai:

```
idaseg@debian:/$ echo $greeting world
hello world
idaseg@debian:/$ bash -c 'echo $greeting world'
world
```

Para remover uma variável, usamos o comando `unset`:

```
idaseg@debian:/$ echo $greeting
hello
idaseg@debian:/$ unset greeting
idaseg@debian:/$ echo $greeting

idaseg@debian:/$
```

OBS: o `unset` requer o nome da variável como argumento. Portanto, não é possível adicionar `$` ao nome, já que isso resolveria a variável e passaria o valor da variável para `unset` em vez do nome da variável.

Variáveis globais

Para disponibilizar uma variável local para subprocessos, podemos transformá-la em variável de ambiente. Isso é possível por meio do comando `export`. Quando executado com o nome da variável, essa variável é adicionada ao ambiente do Shell:

```
idaseg@debian:/$ greeting=hello
idaseg@debian:/$ export greeting
```

OBS: aqui também não se deve usar `$` ao executar `export`, já que queremos transmitir o nome da variável e não seu conteúdo.

Uma maneira mais fácil de criar a variável de ambiente é combinar os dois métodos acima: atribuir o valor da variável na parte de argumento do comando.

```
$ export greeting=hey
```

Vejamos se a variável está acessível aos subprocessos:

```
idaseg@debian:/$ export greeting=hey
idaseg@debian:/$ echo $greeting world
hey world
idaseg@debian:/$ bash -c 'echo $greeting world'
hey world
```

Outra maneira de usar variáveis de ambiente é colocá-las na frente dos comandos. Vamos testar essa possibilidade com a variável de ambiente `TZ`, que contém o fuso horário. Essa variável é usada pelo comando `date` para determinar qual hora do fuso horário deve ser exibida:

```
idaseg@debian:/$ TZ=EST date
sex 06 mai 2022 15:29:47 EST
```

Para exibir todas as variáveis de ambiente, use o comando `env`.

A variável `PATH`

A variável `PATH` é uma das variáveis de ambiente mais importantes de um sistema Linux. Ela armazena uma lista de directórios, separados por dois-pontos, que contêm programas executáveis que funcionam como comandos do Shell do Linux.

```
idaseg@debian:/$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Para acrescentar um novo directório à variável, usamos o sinal de dois-pontos (`:`).

```
$ PATH=$PATH:new_directory
```

Veja um exemplo:

```
idaseg@debian:/$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
idaseg@debian:/$ PATH=$PATH:/home/user/bin
idaseg@debian:/$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/user/bin
```

Como se vê, `$PATH` é usado no novo valor atribuído a `PATH`. Essa variável é resolvida durante a execução do comando e garante que o conteúdo original da variável seja preservado. Obviamente, também dá para usar outras variáveis na atribuição:

```
idaseg@debian:/$ mybin=opt/bin
idaseg@debian:/$ PATH=$PATH:$mybin
idaseg@debian:/$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/user/bin:opt/bin
```

A variável `PATH` deve ser usada com cautela, pois é crucial para o trabalho na linha de comando. Vamos considerar a seguinte variável `PATH`:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Para descobrir como o Shell chama um comando específico, `which` pode ser executado com o nome do comando como argumento. Podemos, por exemplo, tentar descobrir onde o `nano` está armazenado:

```
idaseg@debian:/$ which nano
/usr/bin/nano
```

Neste exemplo, o executável `nano` está localizado no directório `/usr/bin`. Vejamos agora como remover o directório da variável e verificar se o comando ainda funciona:

```
idaseg@debian:/$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin/:/sbin:/bin:/usr/games/
idaseg@debian:/$ echo &PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin/:/sbin:/bin:/usr/games/
```

Vamos procurar pelo comando `nano` novamente:

```
idaseg@debian:~$ which nano: no nano in /usr/local/sbin:/usr/local/bin/:usr/sbin:/sbin:/bin:/usr/gam
es
/usr/bin/nano
```

Como vimos, o comando não foi encontrado e não foi executado. A mensagem de erro também explica o motivo pelo qual o comando não foi encontrado e em quais locais foi buscado.

Vamos adicionar novamente os directórios e tentar executar o comando novamente.

```
idaseg@debian:~$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
idaseg@debian:~$ which nano
/usr/bin/nano
```

Agora o comando voltou a funcionar.

OBS: A ordem dos elementos em `PATH` também define a ordem de pesquisa. O primeiro executável da lista que corresponda às especificações será executado.

Conclusão

Agora conhece as variáveis no Linux e está pronto para um conteúdo mais avançado. Como pode ver, existem variáveis locais estão disponíveis apenas para o processo atual do Shell. Já as variáveis de ambiente estão disponíveis tanto em uma sessão de Shell específica quanto em subprocessos gerados a partir dessa sessão de Shell.

as variáveis não são persistentes. Quando o Shell em que foram configuradas é fechado, todas as variáveis e seus conteúdos são perdidos. A maioria dos Shells oferece arquivos de configuração que contêm variáveis definidas sempre que um novo Shell é iniciado. As variáveis que devem ser definidas permanentemente precisam ser adicionadas a um desses arquivos de configuração.