PROGRAMAÇÃO EM SHELL SCRIPT

UNIDADE 3 - CONTROLE DE FLUXO DA INFORMAÇÃO

EAD.br

Introdução

Olá, caro estudante! Seja bem-vindo a esta unidade de **Programação em Shell Script**. Será que o Linux apresenta boas soluções em TI? Para trabalhar com ele, é necessário ter conhecimento em Shell Script? Sua necessidade ou foco é automatizar tarefas?

Saiba, desde já, que o Linux tem excelentes ferramentas de TI, as quais ajudam a criar e a otimizar soluções, proporcionando à empresa, por exemplo, um sistema que a ajudará na gestão da área da TI. Hoje, muitas empresas adotam o sistema Linux e, por esse motivo, é importante que você invista seus esforços na aquisição desse conhecimento. O Shell Script oferecerá a você eficiência e praticidade, além de ser uma ferramenta de fácil apreensão. É bem possível que você já tenha familiaridade com seus algoritmos e seu mecanismo de funcionamento. Nesse caso, resta apenas aprender sobre sua sintaxe peculiar.

O Shell Script é uma linguagem utilizada em diversos sistemas operacionais, sendo um interpretador de comandos que fica entre o usuário e o kernel Linux. Ele compreende arquivos de texto executáveis, com um ou mais comandos de Shell. Você pode utilizar vários comandos na linha de comandos Shell. Assim, ao encadear uma série de linhas de comandos em um arquivo de texto, você terá um Shell Script, que se utiliza de Interfaces de Linha de Comando (CLI), ou, ainda, Interface Gráfica do Usuário (GUI), para desenvolver aplicações simples ou complexas.

Um script é uma descrição geral de um programa escrito em linguagem interpretada. Esses programas são usados para: simplificar tarefas; substituir comandos executados com frequência, em conjunto com comando único; automatizar a instalação de outros programas; administrar redes de computadores; criar jogos; e executar aplicações interativas.

Como qualquer outra ferramenta, é fundamental conhecer alguns comandos e recursos para trabalhar com Shell Script. Contudo, o mais importante é a criatividade! De fato, muitos jogos eletrônicos são criados com a linguagem Shell, a exemplo de Mario Bros, lançado em 1985 (NINTENDO, 2019). Outras grandes empresas que utilizam Shell Script e servidores Linux são o Google, o Facebook, a Amazon, a Bolsa de Valores de Nova Iorque e até a NASA (INCE, 2015).

Certamente, você já conhece o sistema operacional Linux, certo? Bem como seus fundamentos, características e comandos de gerenciamento? Pois bem, agora é a hora de aprofundar seus conhecimentos sobre o Shell Script. Por isso, ao longo das próximas páginas, abordaremos o controle de fluxo informação, o papel de cada operador de controle e suas respectivas operações.

Fique atento porque o assunto é bem interessante. Bons estudos!

3.1 Fluxos IF e Tabela de Operadores

Saiba que a programação em Shell é, ao mesmo tempo, descomplicada e de alto nível. Não é preciso se preocupar com “o tipo das variáveis, acesso ao hardware, controle de memória, ponteiros, compilação, plataforma, módulos, bibliotecas, bits, bytes, little/big endian” (JARGAS, 2008, p. 26), entre outros fatores complexos. O Shell possui as funcionalidades básicas de uma linguagem estruturada, a integração natural com o sistema operacional e suas ferramentas, as facilidades de redirecionamento, com as possibilidades de combinar vários programas entre si (JARGAS, 2008).

“A programação em Shell é do tipo “lego”, onde a maioria das peças necessárias já existe, bastando saber como combiná-las para produzir soluções. É a filosofia Unix mesclando-se com a arte da programação” (JARGAS, 2008, p. 26). O trabalho com Shell compreende a manipulação de texto e o gerenciamento de processos e de arquivos. As tarefas mais complexas, geralmente, ficam com as ferramentas do sistema, como o grep, sed, dd e find, que se encarregam dos bits e bytes, com interface via linha de comando (JARGAS, 2008). Ficou interessado em aprender mais sobre Shell Script? Então, vamos lá!

3.1.1 Uma pequena introdução

Para começar, saiba que a lógica de programação tem como propósito criar e desenvolver métodos com vistas à solução de problemas, a partir de uma determinada sequência lógica, que fará parte da composição do programa. A lógica é a base para o aprendizado e desenvolvimento das linguagens de programação. A metodologia de encadear ações para alcançar um objetivo precisa ser dominada. Assim, a lógica de programação surge a partir de um conjunto de elementos, tais como organização, perseverança, padronização, criatividade e otimização (ALMEIDA, 2008).

Um programa é um algoritmo criado de acordo com as regras de sintaxe e semântica de uma linguagem de programação. Trata-se de uma sequência de comandos que apresenta as tarefas a serem realizadas com o intuito de solucionar um determinado problema (ALMEIDA, 2008). Se você deseja aprender a linguagem de programação Shell, obrigatoriamente deve aprender a criar algoritmos.

VOCÊ SABIA?

É verdade que "quem não é movido a gasolina, precisa de Shell?" Se você trabalhar em um sistema operacional Unix, a resposta é "Sim!". Ao completar o seu *login*, você já está, inclusive, em um Shell. O Shell é a interface entre o Unix e qualquer agente externo. Em outras palavras, é o programa que lê o comando que você escreveu, convertendo-o em algo mais simples e legível para o Unix. Isso diminuirá o tempo gasto pelo kernel na execução desse comando (NEVES, 2010).

Programar com linguagens de programação tradicionais, tais como C, Pascal e Cobol ou, Java, PHP e Python, é diferente de programar em Shell. Segundo Jargas (2004), a programação é mais tranquila e de alto nível, já que é uma linguagem totalmente interpretada. Um script em Shell é um algoritmo criado com o intuito de executar uma tarefa específica, por meio de comandos do "Bash", além dos executáveis do próprio sistema operacional. O Bash (termo que vem de "*Bourne again Shell*") é o Shell *default* especificado pela configuração inicial. Entretanto, existem outras opções: csh (% C Shell); ksh (Korn Shell); sh ($); zsh (Z Shell); tcsh (%) (ANDRADE et al., 2015).
Clique nos itens e aprenda mais sobre o tema.

Para dar início à escrita de um script, você precisa selecionar um arquivo. Para tanto, clique com o botão direito do mouse no diretório selecionado e, em seguida, selecione uma opção entre duas: "criar novo arquivo de texto" ou "criar novo documento". Lembre-se de que esse procedimento pode ser realizado via terminal também. Nesse caso, digita-se "viexemplo1.sh" (ou "touchexemplo1.sh"). Outro detalhe importantíssimo: para executar um arquivo, é necessário conceder permissão de escrita a ele. O interpretador de comandos do Shell Script, usado para executar e interpretar o script, é passível de definição. Além do Bash, existem, como já mencionamos, o "sh", o "ksh" e o "csh" (NEMETH et al., 2007).

É indispensável que se siga determinado fluxo de execução de tarefas, baseado na necessidade do usuário. É importante ressaltar que os estudos de usuários apresentam, como principais marcos, as reflexões e aplicações concernentes às necessidades e usos de informação, além do processo de satisfação dos usuários como fundamentos para construção de sentidos. Essas construções podem ser norteadas a partir de contemplações dos paradigmas físicos, cognitivos e sócio cognitivos (SILVA, 2012).

O comando que permite essa tomada de decisão é o condicional.

VOCÊ QUER VER?

Python é uma linguagem de programação interpretada e orientada a objetos. Ela foi criada por Guido van Rossum, no início da década de 1990. Hoje, a linguagem Python tem um modelo de desenvolvimento comunitário e aberto, sendo gerenciada pela Python Software Foundation. Seu nome teve inspiração no grupo humorístico inglês Monty Python. Alguns dos filmes do grupo são: "Em busca do cálice sagrado" (1975); A vida de Brian (1979); "O sentido da vida" (1983). Vale a pena conferir!

3.1.2 Fluxos IF

Os interpretadores de linha de comando Shell têm uma longa história. Essa discussão começa com a primeira Shell Unix. Ken Thompson, da Bell Labs, desenvolveu a primeira Shell para Unix em 1971. Chamava-se Shell V6. Era um programa independente do usuário executado fora do kernel. Conceitos como "*globbing*" (reconhecimento de padrões para expansão de parâmetro, como *.txt) foram implementados em um utilitário separado chamado "glob", assim como o comando IF, para avaliar expressões condicionais (JONES, 2011, *online*). O IF é um comando simples, que permite a construção de condicionais de acordo com a seguinte sintaxe:

```
if [ CONDICAO ] ;
then
    AÇÕES
fi
```

Figura 1 - Condicional IF: estrutura de decisão.
Fonte: Elaborada pela autora, 2019.

VOCÊ SABIA?

O vocábulo "script" também é usado na língua portuguesa. Pode fazer referência a um roteiro de peça de teatro, por exemplo. O script seria a descrição da peça, do início ao fim. Na Programação Shell, o conceito de "script" diz respeito a uma lista de comandos a serem executados em sequência. Ou seja, um roteiro predefinido de comandos e parâmetros (JARGAS, 2008).

Ainda em relação ao comando IF, a "condição" verificada será executada caso o teste seja verdadeiro. Nesse caso, passará o controle para o bloco "*then*", dentro do qual as ações serão executadas a partir do comando. Clique nos *cards* e confira algumas dicas importantes (LOUSADA, 2015, *online*).

Dica 01

Para cada "IF" aberto, é necessário um "FI" para fechá-lo.

Dica 02

Para cada "[" aberto, é necessário um "]" para fechá-lo. Lembre-se também de deixar sempre espaços: [CONDICAO]. Os espaços são imprescindíveis, pois servem de atalho para o comando TEST.

Dica 03

Os colchetes poderiam não ser utilizados, dessa forma: if CONDICAO . Esse procedimento assemelha-se ao formato de outras linguagens de programação.

O exemplo a seguir servirá para ilustrar uma aplicação do comando IF (LOUSADA, 2015, *online*):

```
#!/bin/bash
echo "digite um número qualquer:"
read numero;
if [ "$numero" -gt 20 ];
then
    echo "esse número é maior que 20!"
fi
```

Figura 2 - Exemplo de uso do condicional IF.
Fonte: Elaborada pela autora, adaptada de LOUSADA, 2015, *online*.

Há um atalho para o TEST, que seria o uso do comando "[". Nesse sentido, o uso do "[" deixa o IF parecido com o padrão de outras linguagens tradicionais. Veja o exemplo a seguir:

```
if  [ "$VARIAVEL"  -gt 10 ]
then
    echo "é maior que 10"
else
    echo "é menor que 10"
fi
```

Figura 3 - TEST, "[" e o condicional IF.

Fonte: Elaborado pela autora, adaptado de JARGAS, 2004.

Perceba que na maior parte das linguagens de programação, o IF testa uma condição. Contudo, na linguagem Shell, o IF testa a saída de um comando. Observe, a seguir, um código pronto com o condicional IF.

Condicionais com o IF

```
if [ -f "$arquivo" ]; then echo 'Arquivo encontrado'; fi
if [ ! -d "$dir" ]; then echo 'Diretório não encontrado'; fi
if [ $i -gt 5 ]; then echo 'Maior que 5'; else echo 'Menor que 5'; fi
if [$i -ge 5 -a $i -le 10]; then echo 'Entre 5 e 10, incluindo'; fi
if [ $i -eq 5 ]; then echo ' =5'; elif [ $i -gt 5 ]; then echo '>5; else echo '<5'; fi
if [ "$USER" = 'root' ]; then echo '0i root'; fi
if grep -qs 'root' /etc/passwd; then echo 'Usuário encontrado'; fi
```

Figura 4 - Condicionais com o IF.
Fonte: JARGAS, 2008, p. 473.

Os desenvolvedores do Shell introduziram uma sintaxe compacta para redirecionamento (< > e >>) e canalização (| ou ^). Também existe suporte para chamar comandos sequenciais (com ;) e comandos assíncronos (com &). O que faltava no Shell tal qual desenvolvido por Ken Thompson era a capacidade de scripts. Seu único propósito era ser interativo, interpretar comandos para chamar comandos e visualizar resultados (JONES, 2011, *online*).

3.1.3 Operadores

Como você já sabe, os computadores são máquinas programáveis usadas para automatizar tarefas. As ações executadas por ele são baseadas em dados e regras que dependem da aplicação para as quais são criadas. Ele segue as ordens, provenientes de comandos e instruções, contidas em um programa e escritas em uma linguagem de programação. A sequência lógica e coerente dessas ordens é representada por meio de algoritmos, fluxogramas ou diagramas. A implementação dessas ordens em uma linguagem computacional é denominada "programa de computador" (ARAÚJO; LOPES, 2002).

VOCÊ O CONHECE?

O Tux é o mascote oficial do Linux. Ele foi criado por Larry Ewing, em 1996. Seu nome deriva de **T**orvald **U**ni**x**. Um mascote tem sempre forte conexão e expressão com as características da entidade que representa, direcionando a visão de seu público-alvo. De maneira geral, eles trazem elementos primários da identidade visual da empresa: como o símbolo ou logotipo da marca, e cores institucionais.

Figura 5 - Tux, o pinguim mascote do Linux.
Fonte: The Linux Foundation, 2019.

Os algoritmos são uma pseudolinguagem, ou pseudocódigo, e representam as instruções para a solução de um problema com base em um subconjunto de palavras de um sistema linguístico. Eles têm uma estrutura semântica e sintática própria, que pode variar nas suas formas de representação (ARAÚJO; LOPES, 2002).
Um operador é um elemento utilizado nas linguagens de programação, aplicado a um ou mais operandos, em uma operação ou instrução. Os operadores que usam dois operandos são chamados de **operadores binários**. O Shell Script utiliza operadores de forma semelhante às linguagens de programação análogas, como C, Perl, PHP. São exceções os operadores de incremento "+ +" e "- -".
Para usar literalmente um caractere especial sem que o Shell interprete seu significado, você deve colocá-lo entre aspas, ou entre apóstrofos, ou, ainda, inserir uma barra invertida antes dele. Observe o quadro a seguir.

Caractere Especial	Efeito
Aspas	Quando se coloca um caractere especial entre aspas, o Shell ignora seu significado, exceto no caso de um caractere ser um cifrão ($), uma crase (`) ou uma barra invertida.
Apóstrofos	Os apóstrofos são ainda mais restritivos. Nesse caso, todos os caracteres são ignorados.
Barras invertidas	O Shell ignora somente um caractere que segue a barra invertida. Quando colocada ao final da linha, o Shell interpreta a barra invertida como um aviso de continuação da linha, devolvendo um promt secundário, na linha seguinte da tela.

Quadro 1 - Caracteres especiais.
Fonte: Elaborado pela autora, adaptado de NEVES, 2010.

As crases são usadas para avisar ao Shell que aquilo que estiver entre elas é um comando, e para dar prioridade à sua execução. Frequentemente, é necessário priorizar um comando para que o seu resultado seja utilizado por outro. Em virtude do conhecimento matemático, segue-se a tendência de usar os parênteses para priorizar a execução de comandos. Contudo, essa execução é equivocada. No Shell, quando se usa um comando (ou um agrupamento de comandos), o Shell secundário é chamado para executar esses comandos (NEVES, 2010).
A maioria dos comandos tem uma entrada, uma saída e pode gerar erros. A entrada padrão *default* é o teclado do terminal. A saída padrão do comando é a tela do terminal. Para que a execução de comandos não obedeça aos seus respectivos padrões, podemos usar caracteres de redirecionamento. Confira alguns deles a seguir!

Redirecionamento de saída	
>	Redireciona a saída de um comando para um arquivo especificado, inicializando-o, caso não exista ou, ainda, destruindo seu conteúdo anterior.
>>	Redireciona a saída de um comando para um arquivo especificado, anexando-o ao seu fim. Caso o arquivo não exista, ele será criado.
2>	Redireciona os erros gerados por um comando para o arquivo especificado. Mesmo que não ocorra erro na execução do comando, o arquivo será criado.
Redirecionamento de entrada	
<	Avisa ao Shell que a entrada padrão não será o teclado, mas sim o arquivo especificado.
<<	O "here document" serve para indicar ao Shell que o escopo de um comando começa na linha seguinte, e termina quando encontra uma linha cujo conteúdo seja unicamente o label que segue o sinal <<.
Redirecionamentos especiais	
|	O "pipe" serve para direcionar a saída de um comando para a entrada de outro. Normalmente, otimiza a execução do comando.
tee	Captura a saída de um comando com "pipe", copiando o que está entrando no "tee" para a saída padrão e outro comando (ou arquivo).

Quadro 2 - Redirecionamentos de entrada, saída e especiais.
Fonte: Elaborado pela autora, adaptado de NEVES, 2010.

O FTP (*File Transfer Protocol*) é um protocolo que serve para transmitir arquivos entre computadores remotos. Acompanhe o exemplo abaixo:

```
$ ftp -ivn remocomp << FimFTP >> /tmp/$$ 2>> /tmp/$$
> user fulano segredo
> binary
> get arqnada
>FimFTP
$
```

Os sinais > são prompts secundários do UNIX. Enquanto não surgir o label FimFTP o > será o prompt (PS2), para indicar que o comando não Terminou o FTP

Figura 6 - Exemplo usando FTP.
Fonte: NEVES, 2010, p. 98-99.

Descrição

Linha 1:

O trecho "<< FimFTP" avisa ao Shell que, até que o *label* "FimFTP" seja encontrado na coluna 1 de alguma linha, todas as linhas intermediárias pertencem ao comando FTP. E, assim, não deve ser interpretado pelo Shell. A exceção ocorreria se existisse um cifrão ($) desprotegido (sem estar entre apóstrofos ou sem uma barra invertida imediatamente antes) ou um comando entre crases (`...`).

O trecho ">> /tmp/$$" significa que as mensagens do FTP deverão ser anexadas ao arquivo " /tmp/<Num. do Processo>[5] " e "2>> /tmp/$$". Esse procedimento deverá ser feito com as mensagens de erro provenientes do comando.

Linha 2 a 4:

Essas três linhas são o escopo do comando FTP. Nelas, são informados o "LoginName" e o "Password" do usuário. Em seguida, é avisado sobre a transmissão binária (sem interpretação do conteúdo), e é ordenado que seja transmitido o arquivo "arqnada".

Linha 5:

Finalmente, o Shell encontra o término do programa. A partir daí, novamente terá início a interpretação.

Observe outro exemplo, extraído de Neves (2010).

```
$ mail fulano << FimMail                    O mail começa logo após e termina em FimMail.
Sr. Fulano,                                 Observe o comando date entre crases.
Brasil, 'date'                              O 1s -1 está entre crases, logo será resolvido.
Atualmente o conteúdo do meu diretorio eh:  Terminou o texto do mail
'1s -1'
Atenciosamente, EU
FimMail
```

Figura 7 - Exemplo usando e-mail.
Fonte: NEVES, 2010, p. 100.

No exemplo anterior, foi enviado um e-mail para o usuário "fulano". O conteúdo termina quando o Shell encontra o *label* "FimMail" em uma linha. Entretanto, o Shell identificou os comandos "date" e "1s -1" entre crases, e precisou resolvê-los. Dessa forma, a mensagem enviada ao "Sr. Fulano" continha data de envio da correspondência e todos os arquivos que compunham o diretório corrente naquele momento.

VOCÊ SABIA?

Uma fonte de erros comum em scripts é a falta — ou excesso — de espaços em branco, antes ou após um *label*. Fique atento ao fato de que uma determinada linha deve possuir apenas um *label*. Esse tipo de erro é difícil de detectar. Para descobrir espaços e caracteres especiais indesejados, use a instrução CAT, com as opções "– vet" (NEVES, 2010).

Um programa é um conjunto de instruções lógicas que produzem algum resultado quando executadas. Muitas vezes, os dados são fornecidos pelo próprio usuário, e o programa é responsável por prover alguma lógica para o processamento desses dados, para, então, apresentar o resultado desse processamento.

Os **operadores de atribuição**, **aritméticos**, **relacionais** e **lógicos** são muito utilizados na etapa da construção da lógica, possibilitando realizar ações específicas sobre os dados. Adição, subtração, multiplicação e comparação são alguns exemplos.Nesse contexto, a linguagem Shell tem estrutura de programação, uma vez que faz uso de recursos como:

- estruturas de decisão (IF);
- operadores para números;
- operadores para texto;
- operadores lógicos;
- operadores para arquivos e diretórios.

A seguir, confira os operadores aritméticos, de atribuição de valor, de comparação, lógicos e bit a bit, recursos para a criação de variáveis, operações aritméticas, comparações e testes lógicos (JARGAS, 2008).

CASO

Segundo Sergio Prado (2017), Shell Script é uma linguagem de script desenvolvida para o Shell e que muitos acreditam ser limitada. No entanto, ele afirma ser bastante versátil, inclusive no desenvolvimento de jogos eletrônicos. Em seu *site*, Prado disponibiliza uma fase do Mario Bros em Shell Script, bem como o código-fonte, com o qual você poderá interagir e modificar. Disponível em: https://sergioprado.org/desafio-mes-jogando-mario-bros-em-shell-script/. O desenvolvedor de *softwares*, de maneira descontraída, propõe o desafio de alterar o Shell Script a fim de implementar novas funcionalidades no jogo da Nintendo.

VAMOS PRATICAR?

Acesso o *site* de Sergio Prado e siga as instruções a seguir pra implementar duas novas funcionalidades ao tradicional jogo Mario Bros em Shell Script.

1. Ao pressionar CTRL+C durante o jogo, a execução do script é encerrada. Contudo, ele deixa o terminal desconfigurado e a música continua tocando. Seu objetivo é implementar o tratamento do sinal enviado pelo CTRL+C para restaurar a configuração do terminal e parar a música do jogo, caso ela esteja tocando.

2. Assim como Prado, temos certeza que você acha uma injustiça não poder jogar com o Luigi. Seu segundo objetivo, portanto, é adicionar o Luigi como um personagem jogável. Ao executar o script, o jogador poderá selecionar o personagem, passando seu nome como parâmetro, "*./mario.sh mario*", para jogar com Mário, e "*./mario.sh luigi*", para jogar com Luigi.

Desafio aceito?

Lista de operadores

Os operadores aritméticos realizam as operações fundamentais da matemática, operando duas variáveis e retornando o resultado. Para a construção de operações maiores ou mais complexas, é possível combinar esses operadores e criar expressões.

Operadores Aritméticos	
+	adição
-	subtração
*	multiplicação
/	divisão
%	módulo
**	exponenciação

Quadro 3 - Operadores aritméticos.
Fonte: JARGAS, 2008, p. 458.

O operador de atribuição é utilizado para definir o valor inicial ou sobrescrever o valor de uma variável.

Operadores de Atribuição	
=	atribui valor a uma variável
+ =	incrementa a variável por uma constante
- =	decrementa a variável por uma constante
* =	multiplica a variável por uma constante
/ =	divide a variável por uma constante
% =	resto da divisão por uma constante
+ +	incrementa em 1 o valor da variável
- -	decrementa em 1 o valor da variável

Quadro 4 - Operadores de atribuição.
Fonte: JARGAS, 2008, p. 458.

Os operadores relacionais verificam se o valor (ou o resultado) da expressão lógica à esquerda é igual ou diferente ao da direita, retornando um valor booleano. Também definem se o operando à esquerda é menor, menor ou igual, maior ou maior ou igual ao da direita, retornando um valor booleano (JARGAS, 2008).

Operadores Relacionais	
= =	igual
! =	diferente
>	maior
> =	maior ou igual
<	menor
< =	menor ou igual

Quadro 5 - Operadores relacionais.
Fonte: JARGAS, 2008, p. 458.

Os **operadores lógicos** permitem criar expressões lógicas maiores a partir da junção de duas ou mais expressões. Para tanto, aplicam-se as operações lógicas "E" (representado por "&&") e "OU" (representado por "||").

Operadores Lógicos	
&&	E lógico (AND)
||	OU lógico (OR)

Quadro 6 - Operadores lógicos.
Fonte: JARGAS, 2008, p. 458.

Ainda há os operadores "bitwise", que servem para alterar a sequência de bits de uma variável.

Operadores de BIT	
<<	deslocamento à esquerda
>>	deslocamento à direita
&	E de bit (AND)
|	OU de bit (OR)
^	OU exclusivo de bit (XOR)
~	negação de bit
!	NÃO de bit (NOT)

Quadro 7 - Operadores de BIT.
Fonte: JARGAS, 2008, p. 458.

Os operadores "<<" fazem o deslocamento dos bits para a esquerda, e preenchem o restante com 0. Os operadores ">>" fazem o deslocamento dos bits para a direita, e preenchem o restante com 0. Já o operador "&" compara os bits de cada variável, um por um. Caso dois bits (um da variável "a" e outro da variável "b") sejam iguais a 1 (bit ligado), o retorno é 1. Caso contrário, o retorno é 0. O operador "|" também compara os bits de cada variável, um por um. Quando pelo menos um dos bits é igual a 1, o retorno é 1. Caso contrário, o retorno é 0. O operador "^", por sua vez, compara os bits, de forma que se os 2 bits (um da variável "a" e outro da variável "b") forem iguais, ele retorna 0. Caso contrário, ele retorna 1. Por fim, o operador "~" inverte os bits de uma variável: onde era 1 fica 0, e onde era 0 fica 1.

Operadores de BIT (atribuição)	
<< =	deslocamento à esquerda
>> =	deslocamento à direita
& =	E de bit
| =	OU de bit
^ =	OU exclusivo de bit

Quadro 8 - Operadores de BIT (atribuição).
Fonte: JARGAS, 2008, p. 459.

O uso indevido de operadores é uma das causas mais comuns de erros na hora de programar. Por isso, fique atento às regras.

VOCÊ QUER LER?

Você, estudante de um curso de tecnologia, já parou para pensar na importância de problematizar suas aplicações no contexto social? O filme Matrix traz em seu enredo a descrição de um futuro distópico, no qual a realidade é simulada, criada por máquinas inteligentes (Inteligência Artificial), e usada para subjugar a população humana, enquanto o calor de seus corpos é usado como fonte de energia. O livro que inspirou a trilogia dos irmãos Wachowski chama-se "Simulacros e Simulação", do filósofo francês Jean Baudrillard. Sugerimos a leitura!

O Shell é uma linguagem de programação completa, ou seja, de quarta geração (4GL). É considerada uma ferramenta original e de protótipo rápido. Isso significa dizer que ela é capaz de ensinar conceitos-chaves, como modularidade, reutilização e desenvolvimento. Além disso, o Shell Script dispõe de variáveis, construções condicionais, interativas e ambiente adaptável ao usuário.

O Linux é o exemplo mais proeminente de sistemas operacionais de código aberto e *software* livre. Seu código-fonte pode ser usado, modificado e distribuído. Sobre o Shell Script, um interpretador de comandos Linux, várias são as suas aplicações e, consequentemente, os segmentos de empresa públicas e privadas que fazem uso de suas tecnologias.

Como já dito, o Linux é o núcleo ou kernel do sistema operacional. Ele é de uso gratuito. Na década de 1990, foi distribuído sob licença de *software* gratuita, cujo o nome era GNU (*General Public License* – GLP). Nos últimos anos, muitos usuários, empresas, corporações, universidades sem fins lucrativos de diversos países, têm aderido ao Linux.

Listamos, a seguir, as vantagens de se usar Linux, segundo Ball e Duff (2004). Clique e acompanhe!

Vantagem 01

O Linux tem um grande retorno sobre o investimento. Há pouco (ou nenhum) gasto com máquina, ao contrário dos sistemas operacionais comerciais. Não tem taxas de royalties ou licenciamento. Uma cópia de distribuição pode formar a base de uma distribuição de *software* empresarial, com aplicativos de *software* de produtividade. Esse recurso pode simbolizar uma significativa economia nos custos de serviço e tecnologia da informação.

Vantagem 02

O Linux funciona bem no *desktop*. A interface tem funcionado bem como sistema operacional Unix. Além disso, o Linux tem funcionado como uma plataforma de servidor. Ele é rápido, seguro, estável, escalável e robusto. As últimas versões do kernel do Linux suportam computadores com vários processadores, ampla memória, arquivos individuais grandes, diversas opções de modernos sistemas de arquivos, com *journaling*, vários utilitários de monitoramento e controle de processos. O Linux tem custo de entrada e implementação baixo, e os custos de manutenção também podem ser reduzidos.

Vantagem 03

O Linux desperta interesse de um amplo público do setor de *hardware* e *software*. Há versões de Linux para todos os tipos de CPUs. Além disso, o sistema oferece uma plataforma isenta do pagamento de royalties para a programação. Em função do desenvolvimento de código-fonte aberto e à disponibilidade de ferramentas de desenvolvimento de alta qualidade gratuitas, o Linux não exige investimentos altos de desenvolvedores iniciantes, bem como de novas indústrias de tecnologia.

Vantagem 04

O suporte de empresas com notoriedade no mercado de *hardware*, como a IBM, atribui credibilidade ao Linux. A IBM suporta o Linux em toda a sua linha, desde os *notebooks* até os computadores de grande porte. Por isso, novos clientes corporativos têm aderido às soluções Linux.

O licenciamento do *software* é uma questão importante para todos os usuários, já que pode acarretar em considerações morais, legais e financeiras. A End User Licenced Agreement (EULA) reconhece que você adquiriu apenas o direito de usar o *software*, de acordo com termos específicos. "A paixão [...] move o mundo do *software* livre, um espírito de colaboração que transformou o Linux no maior fenômeno no mundo dos sistemas operacionais desde a criação do Windows". Essa foi a chamada da matéria Linux (O que Você Pode Ganhar ou Perder com a Revolução do Pinguim?), que saiu na revista técnica InfoExame, uma espécie de "termômetro" do que está acontecendo no mundo da tecnologia, em 2002 (APGAUA, 2004).

Clique nas setas e conheça mais sobre o tema.

"O sistema Linux seria, assim, o precursor de uma revolução ou um fenômeno mundial?", pergunta Apgaua (2004, p. 223). De maneira geral, alguns pressupostos sustentam o código-fonte aberto. Para começar, o sistema operacional Linux é livre, como você já sabe. Qualquer pessoa pode melhorar seu código-fonte (as instruções de programação implícitas no sistema). Tais experimentos precisam ser disponibilizadas livremente.

Pense Zen: essa é a filosofia Linux! O projeto não pertence a ninguém e pertence a todos. Por isso há um aperfeiçoamento rápido e contínuo. Com colaboradores trabalhando em paralelo, os resultados podem acontecer muito mais depressa e com muito mais sucesso do que se estivessem sendo conduzidos a portas fechadas (TORVALDS; DIAMOND, 2001, apud APGAUA, 2004, p. 223).

Essa forma de produção de *software* cooperativa, descentralizada e "anárquica" foi chamada por Eric S. Raymond, de "método bazar" (gratuito ou de baixo custo), como contraponto ao "método catedral", "forma centralizada e controlada de se desenvolver *software*" e que "necessita de um arquiteto central" (GONÇALVES JR; SILVA, 1999, apud APGAUA, 2004, p. 223).

VAMOS PRATICAR?

Agora você já sabe que o Shell Script é um interpretador de comandos que fica entre o usuário e o kernel Linux. Ele compreende arquivos de texto executáveis, com um ou mais comandos de Shell. Tais programas são usados para simplificar tarefas, substituir comandos executados com frequência, automatizar a instalação de outros programas, administrar redes de computadores, criar jogos e executar aplicações interativas. Utilizam CLI ou GUI, o que permite que sejam desenvolvidas aplicações simples ou complexas. Assim, responda:

- O que são operadores? O Shell Script utiliza operadores?
- Para que serve o uso das aspas ("...") em Shell Script?
- A linguagem Shell tem estrutura de programação? Justifique sua resposta.

Síntese

Nesta unidade, você aprendeu sobre Shell Script, interpretador de comandos que fica entre o usuário e o kernel Linux. O Shell Script é utilizado para simplificar, substituir, automatizar, otimizar, criar e executar tarefas, comandos, instalações e aplicações interativas.

Nesta unidade, você teve a oportunidade de:

- conhecer o comando que permite a tomada de decisão, e de maneira simples, que é o condicional;
- entender que os algoritmos são pseudolinguagem, ou pseudocódigo;
- entender que um operador é um elemento utilizado nas linguagens de programação, aplicado a um ou mais operandos, em uma operação ou instrução;
- compreender que o Shell Script utiliza operadores de forma semelhante às linguagens de programação C, Perl, PHP;
- compreender que o Shell Script utiliza operadores.

Bibliografia

ALMEIDA, M. **Curso essencial de lógica de programação**. São Paulo: Digerati Books, 2008.

ANDRADE, A. V. et al. **Linux**: comandos básicos e avançados. Diamantina, 2015.

APGAUA, R. O Linux e a perspectiva da dádiva. **Horizontes Antropológicos**, Porto Alegre, ano 10, n. 21, p. 221-240, jan./jun. 2004.

ARAÚJO, L. P.; LOPES, M. C. 2002. **Introdução a programação**. Blumenau: Universidade Regional de Blumenau, 2002.

A VIDA de Brian. *Monty Python's Life of Brian*. Direção: Terry Jones. Produção: Python (Monty) Pictures, HandMade Films. Reino Unido. Duração: 93min.

BALL, B.; DUFF, H. **Dominando Linux, Red Hat e Fedora**. São Paulo: Pearson, 2004.

BAUDRILLARD, J. **Simulacros e Simulação**. Lisboa: Relógio d'Água, 1991.

EM BUSCA do cálice sagrado. *Monty Python and the Holy Grail*. Direção: Terry Gilliam, Terry Jones. Produção: Python (Monty) Pictures. Reino Unido. Duração: 92min.

JARGAS, A. M. **Introdução ao Shell Script**. 2004. Versão *online*. Disponível em: https://aurelio.net/shell/apostila-introducao-shell.pdf. Acesso em: 30/09/2019.

JARGAS, A. M. **Shell Script Profissional**. São Paulo: Novatec, 2008.

JONES, M. T. **Evolução de Shells no Linux**: de Bourne a Bash, e além, 22 dez. 2011. Disponível em: https://www.ibm.com/developerworks/br/library/l-linux-shells/index.html. Acesso em: 30/09/2019.

LOUSADA, F. **Introdução ao Shell Script no Linux**. 2015.

NEMETH, E et al. **Manual Completo do Linux**. São Paulo: Pearson Prentice Hall, 2007.

NEVES, J. C. **Programação Shell Linux**. 8. ed. Rio de Janeiro: Brasport, 2010.

NINTENDO. History. Disponível em: https://mario.nintendo.com/history/. Acesso em: 30/09/2019.

O SENTIDO da vida. *Monty Python's The Meaning of Life*. Direção: Terry Jones. Produção: Celandine Films, The Monty Python Membership. Reino Unido. Duração: 90min.

SERGIOPRADO.ORG. [Desafio do mês] **Jogando Mario Bros em Shell Script**, 19 jun. 2017. Disponível em: https://sergioprado.org/desafio-mes-jogando-mario-bros-em-shell-script/. Acesso em: 30/09/2019.

THE LINUX FOUNDATION. Homepage. Disponível em: https://www.linuxfoundation.org/. Acesso em: 30/09/2019.

UNIDADE DE TECNOLOGIA E INFORMAÇÃO. **30 grandes empresas, dispositivos e locais que rodam Linux**. Santa Maria: UFSM, 2015. Disponível em: http://coral.ufsm.br/unitilince/index.php/2015-11-13-10-48-41/227-30-grandes-empresas-dispositivos-e-locais-que-rodam-linux. Acesso em: 30/09/2019.