COMPREENDENDO O COMPORTAMENTO DO SHELL SCRIPT, PARTINDO DE UMA ANÁLISE COMPARATIVA

Uma linguagem de programação é selecionada de acordo com a compatibilidade de projetos que serão empregados. Por esse motivo, é necessário que se tenha conhecimento mínimo poder avaliar e escolher. Os algoritmos produzidos nas linguagens C, Java e Python buscam solucionar problemas clássicos computacionais (DRUMMOND; GOMES; SEABRA, 2018). 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 (JARGAS, 2004). Para escrever um script, é necessário selecionar um arquivo. Primeiramente, você clica com o botão direito do mouse no diretório selecionado e, em seguida, escolhe entre “criar novo arquivo de texto” ou “criar novo documento”. Esse procedimento pode ser realizado via terminal também. Para tanto, digite “vi exemplo1.sh” (ou “touch exemplo1.sh”). Para que um arquivo possa ser executado, você deverá conceder permissão de escrita a ele. O interpretador de comandos do Shell Script, usado para executar e interpretar o script, é passível de definição.

Frequentemente, é indispensável seguir determinado fluxo de execução de tarefas, que é baseado na necessidade do usuário. O comando que permite essa tomada de decisão, de maneira simples, é o condicional. O “IF” é um comando simples, que permite a construção de condicionais, de acordo com a seguinte sintaxe:

if [ CONDICAO ] ;

then

AÇÕES

fi

A “condição” verificada será executada caso o teste seja verdadeiro. Se sim, passará o controle para o bloco “then”, dentro do qual as ações serão executadas a partir do comando. O exemplo a seguir servirá para ilustrar uma aplicação do comando “IF” (LOUSADA, 2015):

#!/bin/bash

echo “digite um número qualquer:”

read numero;

if [ “$numero” –gt 20 ];

then

echo “esse número é maior que 20!”

fi

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:

if [ “$VARIAVEL” -gt 10 ]

then

echo “é maior que 10”

else

echo “é menor que 10”

fi

Nota 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.

Referências

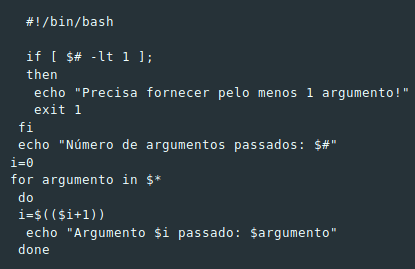
DRUMMOND, I. N.; GOMES, F. C.; SEABRA R. D. Análise Comparativa de Linguagens de Programação a partir de Problemas Clássicos da Computação. Revista de Sistemas e Computação, Salvador, v. 8, n. 1, p. 56-76, jan./jun. 2018. Disponível em: <https://revistas.unifacs.br/index.php/rsc/article/view/5133>. Acesso em: 08/10/2019.

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.

LOUSADA, F. Introdução ao Shell Script no Linux. 2015. Disponível em: <https://www.devmedia.com.br/introducao-ao-shell-script-no-linux/25778>. Acesso em: 08/10/2019.

Vamos praticar

O exemplo a seguir traz uma aplicação do código Shell. Observe seu comportamento, ou seja, suas linhas e comandos:



Nesta atividade, você deverá pesquisar códigos relacionados a operações matemáticas simples e problemas complexos e escrever a solução em Shell Script. Sua descrição deverá ser minuciosa, com explicações dadas linha à linha de código entregue.

Não se esqueça de disponibilizar suas conclusões no fórum da seção “Compartilhe”.

Código escrito em Shell Script disponível em: [praticaUnidade3GRA0946](praticaUnidade3GRA0946.sh)

FEEDBACK

Caro estudante,

nos estudos computacionais que envolvem linguagens de programação, muito se discute sobre qual a melhor opção para se utilizar em um projeto específico ou qual linguagem possui melhor desempenho em certa aplicação. Análises qualitativa e quantitativa das linguagens podem servir de parâmetros de tempo de execução, quantidade de linhas de código, tamanho do arquivo, tempo de produção, quantidade de acessos à documentação e uso de funções nativas.

Referência

DRUMMOND, I. N.; GOMES, F. C.; SEABRA R. D. Análise Comparativa de Linguagens de Programação a partir de Problemas Clássicos da Computação. Revista de Sistemas e Computação, Salvador, v. 8, n. 1, p. 56-76, jan./jun. 2018. Disponível em: https://revistas.unifacs.br/index.php/rsc/article/view/5133. Acesso em: 08/10/2019.