SHELL SCRIPT
PROFISSIONAL

Aurélio Marinho Jargas

Novatec

Capítulo 1

Programas sim, scripts não

Este livro ensina a fazer programas e não scripts. Seu objetivo é transformar "scripteiros" em programadores, dando o embasamento necessário e ensinando as boas práticas da programação. Isso melhorará a qualidade do código, facilitando muito o trabalho de manutenção futura. Chega de reescrever scripts porque o código original estava incompreensível, chega de dores de cabeça pela falta de limpeza e organização. Aprenda a fazer programas em shell, do jeito certo.

Em geral, qualquer código feito em shell é logo chamado de script, não importando seu tamanho, sua qualidade ou sua função. Mas há uma grande diferença entre um script e um programa.

O que é um script?

A palavra script também é usada em português para referenciar o roteiro de uma peça de teatro ou produção televisiva. Esse script é o caminho que os atores devem seguir, a descrição de todo o show do início até o fim.

Em programação shell, um script não é muito diferente disso. É uma lista de comandos para serem executados em seqüência (um comando após o outro). Um roteiro predefinido de comandos e parâmetros. Geralmente o que está no script é aquilo que o usuário digitaria na linha de comando. Realmente, apenas colocando-se todo o histórico da linha de comando em um arquivo, tem-se um script!

Uma curiosidade do script é que as pessoas parecem não levá-lo muito a sério. Fazem códigos feios, sem comentários, sem alinhamento, sem muita clareza. Parece que scripts são sempre feitos "nas coxas" e por quem não têm muita noção de programação. No fim, este acaba sendo um grande trunfo da programação em shell: qualquer um pode fazer scripts! Basta saber usar a linha de comando e pronto, nasceu mais um scripteiro. Por isso, o shell é tão difundido e popular para automatização.

Como o shell é poderoso e gostoso de usar, com o tempo mais e mais tarefas começam a ser desempenhadas por scripts e estes começam a crescer. Logo começam a ficar lentos, complicados, difíceis de manter. Aquela brincadeira de programar de maneira desleixada, agora tornou-se uma maçaroca sem início nem fim.

É normal também escrever scripts rápidos, de poucas linhas, que são usados por alguns dias e depois simplesmente esquecidos. Como é fácil perder scripts! Eles são descartáveis. Mas tudo bem, isso não é um problema desde que se tenha em mente que o script recém-escrito não era para ser sério. O problema é quando um script desses é o responsável por uma tarefa importante, vital para o funcionamento de uma empresa, por exemplo.

Os scripts são ótimos para automatizar tarefas repetitivas e um lugar propício de elas acontecerem é no servidor. Os administradores de sistema sempre acabam recorrendo aos scripts para não precisarem fazer manualmente aquelas tarefas chatas, rotineiras. Só que administradores em geral não são programadores, então o resultado é um emaranhado de pequeninos scripts de "5 minutos" que mantêm serviços importantes funcionando, monitoram conexões, fazem becape... Já imaginou que seus dados importantes podem estar passando por um destes scripts?

O que é um programa?

Há várias definições que podem ser usadas para descrever um programa. Em nosso contexto, um programa é um **script feito do jeito certo**.

Programas não são feitos nas coxas. Eles são pensados, analisados, codificados com cautela, têm comentários, cabeçalho, tratam erros e exceções, são alinhados, bonitos de se ver. Um programa não é uma maçaroca, não é enigmático e tampouco descartável. Um programa é feito para ser funcional, eficiente e principalmente: é feito para ser atualizado.

Um programa não é uma obra de arte imutável. Pelo contrário! Um programa é vivo, mutante, imperfeito, incompleto, que está sempre melhorando, ficando menor/maior, mais rápido/lento. É raro encontrar programas "versão final", em que não há mais o que melhorar. Além da simples automatização de tarefas, programas são feitos para resolver problemas e suprir necessidades. Problemas mudam, necessidades mudam, então programas também mudam!

Um programador não é um scripteiro. Ele reconhece a importância de um código limpo e legível, que facilite o trabalho de manutenção e compreensão de seu funcionamento. De que adianta fazer um código complicadíssimo, rápido e inovador se na hora de atualizá-lo ninguém souber como alterá-lo e não restar outra solução senão o famoso reescrever do zero?

Principais diferenças entre scripts e programas

Script	Programa
Codificação descuidada	Codificação cautelosa
Código feio e sem estrutura	Código limpo
Pouca manutenção, descartável	Vivo, evolução constante
Feito por um usuário	Feito por um ou mais programadores
Bugs são ignorados	Bugs são encontrados e corrigidos

Por que programar em shell?

O primeiro passo de quem aprende shell é fazer scripts. Administradores de sistemas precisam fazer scripts para automatizar tarefas do servidor, usuários fazem scripts para aprender a programar ou para criar pequenas ferramentas de auxílio. Fazer scripts é fácil e tranqüilo, com poucas linhas é possível desfrutar das vantagens que a automatização e a padronização nos trazem.

Já programas são um passo maior. É preciso tempo de dedicação e estudo para resolver problemas e codificar soluções. Mas, se existem várias linguagens de programação mais poderosas e flexíveis que o shell, então por que usá-lo para fazer programas? Simples: porque o tempo de aprendizado é reduzido.

Um ponto-chave é o conhecimento prévio em shell script. Se a pessoa investiu seu tempo em aprender os comandos e estruturas do shell, já sabe fazer seus scripts e sente-se à vontade com a linguagem, por que não dar um passo maior e fazer programas completos? Não se sai do zero, e sim se aperfeiçoa um conhecimento que já possui. Com isso, o caminho a percorrer é menor e em pouco tempo o antigo scripteiro poderá estar fazendo programas com P maiúsculo.

Programadores de outras linguagens que são novatos no shell também se beneficiarão do aprendizado rápido. A parte mais difícil que são os algoritmos e os mecanismos de funcionamento de um programa já são conhecidos. Resta apenas aprender a sintaxe e as características do shell, que são facilitados por serem aprendidos diretamente na linha de comando do sistema operacional.

Programar em shell é diferente!

Para quem conhece linguagens de programação tradicionais como C, Pascal e Cobol, ou as mais moderninhas como Java, PHP, Python e Ruby, logo notará que programar em shell é diferente.

A programação é mais tranqüila, 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 e outros complicadores. Para o programador, resta a parte boa: algoritmos. Ele fica livre para criar soluções e deixa de se preocupar com limitações da máquina ou da linguagem.

Programar em shell geralmente envolve a manipulação de texto e o gerenciamento de processos e de arquivos. As tarefas complexas ficam com as ferramentas do sistema como `grep`, `sed`, `dd` e `find` que se encarregam dos bits e bytes e possuem interface amigável via opções de linha de comando.

Além de possuir as funcionalidades básicas de uma linguagem estruturada normal e a integração natural com o sistema operacional e suas ferramentas, há as facilidades de redirecionamento, em que é possível combinar vários programas entre si, multiplicando seus poderes e evitando reescrita de código. Pelo uso intensivo dessas ferramentas e da possibilidade de interoperabilidade entre elas, a programação em shell é chamada do tipo LEGO, onde a maioria das peças necessárias já existe, bastando saber como combiná-las para produzir soluções.

É a filosofia do Unix mesclando-se com a arte da programação.