

TRANSMISSÃO DE DADOS VIA REDE ELÉTRICA

Conheça as vantagens e os
desafios para sua adoção



VOCÊ É ASSINANTE MVP?

ENTÃO PODE TER CERTEZA:
O MERCADO
PREFERE VOCÊ!



DESENVOLVEDOR ATUALIZADO É O MAIS VALIOSO DO MERCADO!

Muitos profissionais estão acomodados, não se atualizam, não leem livros e não vão a eventos. Achem que é a empresa que deve mantê-los atualizados. Gastam dinheiro com tudo, mas acham caro investir menos de R\$60,00 por mês em suas carreiras.

Mas você não! Você é MVP! Parabéns!



QUEM TEM ESTÁ TRANQUILO.



TENHA ACESSO A:



+DE **260** CURSOS ONLINE



09 REVISTAS MENSAIS



7.850 VÍDEO-AULAS

POR
APENAS **59,90**
MENSAIS

Acesse: www.devmedia.com.br/mvp

Sumário

06 – Introdução ao Shell Script

[Ivani Nascimento e Pathiene Gerstenberger]

18 – Transmissão de dados via rede elétrica

[Christiane Borges Santos]

Solução completa, Curso

24 – Migração de dados: metodologias e abordagens host-based – Parte 2

[Telmo Eduardo Nóbrega Reis]

FÓRUM DEVMEDIA

O lugar perfeito para você ficar por dentro de tudo o que acontece nas tecnologias do mercado atual



No fórum da DevMedia você irá encontrar uma equipe disponível e altamente qualificada com consultores e colaboradores prontos para te ajudar a qualquer hora e sobre qualquer assunto. Temos as salas de Java, .NET, Delphi, Banco de Dados, Engenharia de Software, PHP, Java Script, Web Design, Automação comercial, Ruby on Rails e muito mais!

ACESSE AGORA
www.devmedia.com.br/forum



Edição

Editor

Eduardo Spínola (eduspinola@gmail.com)

Sub Editores

Marco Antônio Pereira Araújo (maraujo@devmedia.com.br)

Rodrigo Oliveira Spínola (rodrigo.devmedia@gmail.com)

Consultora Técnica

Daniella Costa (daniella@devmedia.com.br)

Produção

Jornalista Responsável Kaline Dolabella - JP24185

Diagramação e Capa Romulo Araujo

Atendimento ao leitor

A DevMedia possui uma Central de Atendimento on-line, onde você pode tirar suas dúvidas sobre serviços, enviar críticas e sugestões e falar com um de nossos atendentes. Através da nossa central também é possível alterar dados cadastrais, consultar o status de assinaturas e conferir a data de envio de suas revistas. Acesse www.devmedia.com.br/central, ou se preferir entre em contato conosco através do telefone 21 3382-5038.

Publicidade

publicidade@devmedia.com.br – 21 3382-5038

Anúncios – Anunciando nas publicações e nos sites do Grupo DevMedia, você divulga sua marca ou produto para mais de 100 mil desenvolvedores de todo o Brasil, em mais de 200 cidades. Solicite nossos Media Kits, com detalhes sobre preços e formatos de anúncios.

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique a vontade para entrar em contato com os editores e dar a sua sugestão! Se você estiver interessado em publicar

um artigo na revista ou no site Easy Java Magazine, entre em contato com o editor, informando o título e mini-resumo do tema que você gostaria de publicar:

Eduardo Spínola - Editor da Revista
eduspinola@gmail.com

Editorial

A primorizar os meios de transmissão de dados é uma busca constante para profissionais/pesquisadores da área de infraestrutura. Neste cenário, variadas opções surgem a cada ciclo tecnológico. Alguém se lembra dos cabos coaxiais, e da revolução da fibra ótica? Recentemente, uma não tão nova, mas promissora possibilidade de transmissão de dados voltou a ganhar destaque: a transmissão de dados via rede elétrica. É sobre esta rica e interessante opção que destacamos como matéria de capa dessa edição.

O artigo **Transmissão de dados via rede elétrica** descreve conceitos fundamentais sobre as tecnologias PLC (Power Line Communications), HomePlug e Smart Grid, e é motivado por um crescente interesse na aplicabilidade das linhas de energia como meio alternativo de propagação de sinais de comunicação. Com o desenvolvimento da tecnologia PLC, que permite transmissões de sinais elétricos e de dados em redes de distribuição de energia, surge mais uma opção de conectividade em banda larga, além dos sistemas existentes, como wireless, de satélite e cabos coaxiais das operadoras de TV por assinatura.

Introdução ao Shell Script tem por objetivo introduzir fundamentos de Shell script aos administradores de sistema, possibilitando a automatização de tarefas repetitivas e até mesmo a personalização do ambiente. O shell script não é exatamente uma linguagem de programação, mas tem a vantagem de ser mais simples, pois trabalha com uma gama de utilitários para tarefas como consulta de informações, manipulação de textos, elaboração de relatórios, envio de e-mails, e assim por diante. Em razão da diversidade de solução para um mesmo problema, e sendo a criatividade o limite para elaborar Shell script, este artigo não cobre todos os tópicos relacionados ao assunto. Serão apresentadas definições do que é necessário saber para criação de scripts, permitindo que o leitor esteja apto para criar seus próprios programas.

Por fim, em **Migração de dados: Metodologias e abordagens host based – Parte 2** serão apresentados os procedimentos operacionais para efetuar a migração dos dados utilizando as técnicas definidas na primeira parte deste artigo. Utilizaremos recursos do próprio gerenciador de volumes LVM e ferramentas nativas do sistema operacional Linux como RSYNC e TAR.

Eduardo Oliveira Spínola
eduspinola@gmail.com
twitter.com/Java_Magazine



Seu conhecimento tem visibilidade na DevMedia

Publique artigos e vídeos no site e ganhe mais visibilidade em seu trabalho. Somos o maior site do Brasil para desenvolvedores.

Para publicar é bem simples: mande um email para nosso editor **Wesley Yamazack (yamazack@gmail.com)**.



Para mais informações acesse:
**[http://www.devmedia.com.br/
seja-um-autor-devmedia/28103](http://www.devmedia.com.br/seja-um-autor-devmedia/28103)**



Introdução ao Shell Script

Aprenda os conceitos básicos da programação em Shell utilizando o bash para criar scripts e automatizar tarefas

Todo administrador de sistema, programador ou até mesmo usuário final, um dia se depara com a necessidade de criar um Shell script simples (ou não) para economizar tempo e esforço, ou ainda automatizar uma tarefa repetitiva, que pode ser desde um alias (apelido) para um comando utilizado com frequência até uma tarefa mais complexa, como extração de dados para criação de relatórios.

Mas o que é exatamente o Shell? Todo Sistema Operacional (SO) possui um shell, que interpretará os comandos do usuário, como copiar ou remover arquivos a partir de uma interface gráfica (GUI – *Graphical User Interface*) utilizando o mouse, ou a partir do modo linha de comando (CLI – *Command-Line Interface*) em um terminal. Assim, o Shell, antes de tudo, é uma interface para comunicação entre o usuário e o sistema operacional, recebendo como *input* (entrada) comandos que são convertidos em instruções que o SO entenda e execute. Quando um comando é emitido, o shell passa essa solicitação para o kernel traduzir em linguagem apropriada à plataforma que está sendo utilizada. Dessa forma, o sistema tem um maior controle sobre o hardware, impedindo acessos indevidos por parte dos usuários.

Além de shells de sistema operacional baseados em texto, é possível encontrar shells para interface gráfica (como o Windows Explorer no Windows, ou o Nautilus no Linux), shells para linguagens (como Perl/Python shell), ou ainda shells para fins específicos, como o goosh, por exemplo, que funciona como um front-end para navegação web. Com ele é possível realizar pesquisa via linha de comando com o Google por meio de comandos como *search*, *more* e *go*. A **Figura 1** apresenta um exemplo de uso do goosh.

Resumo DevMan

Introdução ao Shell Script:

Este artigo tem por objetivo introduzir fundamentos de Shell script aos administradores de sistema, possibilitando a automação de tarefas repetitivas e até mesmo a personalização do ambiente. O shell script não é exatamente uma linguagem de programação, mas tem a vantagem de ser mais simples, pois trabalha com uma gama de utilitários para tarefas como consulta de informações, manipulação de textos, elaboração de relatórios, envio de e-mails, e assim por diante. Em razão da diversidade de solução para um mesmo problema, e sendo a criatividade o limite para elaborar Shell script, este artigo não cobre todos os tópicos relacionados ao assunto. Serão apresentadas definições do que é necessário saber para criação de scripts, permitindo que o leitor esteja apto para criar seus próprios programas.

Em que situação o tema é útil:

Através dos tempos, o homem sempre tentou resolver seus problemas com a maior rapidez e com o menor esforço possíveis. A busca pela automação de tarefas repetitivas possibilitou a invenção de instrumentos como a roda, a alavanca, o carro e até o computador. Entretanto, as tarefas que devem ser realizadas pelo computador de forma automática têm de ser planejadas e colocadas em uma sequência de ações para serem posteriormente executadas. A vantagem no Linux é que praticamente tudo pode ser automatizado por meio de script, utilizando shells nativos no sistema operacional, onde é possível criar desde rotinas simples para backup até programas que realizam extração de dados para serem utilizados na confecção de relatórios. Seja você um administrador de sistemas, programador ou usuário final, em algum momento irá se deparar com situações em que um simples shell script será utilizado para economia de tempo ou, ainda, para automatizar alguma tarefa importante.

```

Goosh goosh.org 0.5.0-beta #1 Mon, 23 Jun 08 12:32:53 UTC, Google/Ajax

Welcome to goosh.org - the unofficial google shell.

This google-interface behaves similar to a unix-shell.
You type commands and the results are shown on this page.

goosh is powered by google.

goosh is written by Stefan Grothkopp <grothkopp@gmail.com>
it is NOT an official google product!
goosh is open source under the Artistic License/GPL.

Enter help or h for a list of commands.

Loading local settings...

guest@goosh.org:/web> web goosh
1) goosh.org - the unofficial google shell.
   goosh is a google-interface that behaves similar to a unix-shell.
   http://goosh.org/

```

Figura 1. Shell goosh: exemplo de pesquisa Google via linha de comando

O Unix popularizou a separação do shell (parte do sistema que permite que os comandos sejam digitados e interpretados) de todo o resto: o sistema de entrada/saída, gerenciamento de memória/CPU, entre outras tarefas que o SO executa para que o usuário não tenha que se preocupar. Mas esse conceito foi o início de uma revolução, na qual o shell era mais um programa que rodava no Unix e contava com pelo menos dois tipos: o Bourne Shell (sh), descendente do Thompson Shell (original do Unix, que possuía estruturas básicas de controle e era minimalista a ponto de as declarações *if* e *goto* serem implementadas como comandos externos) e o C Shell (csh) desenvolvido por Billy Joe da Universidade de Berkeley, que tinha por objetivo criar uma linguagem de script que fosse semelhante à linguagem C. Ao longo dos anos, outras alternativas surgiram, como o Korn Shell e o Bourne Again Shell, que será utilizado neste artigo.

Quando se fala em Shell, geralmente imagina-se apenas um terminal complexo que aceita comandos, porém o shell faz mais que isso: é possível ter um histórico dos comandos digitados que podem ser editados e reutilizados, personalizar o ambiente definindo comandos próprios (criação de alias), criar atalhos, etc. E, como se não fosse o bastante, o shell também é programável, possibilitando que sequências de comandos digitados repetidamente possam ser colocadas em um arquivo e transformadas em um programa, comumente chamado de script. Compactar arquivos de log do sistema para backup, renomear uma lista de arquivos, localizar arquivos criados há mais de 30 dias no disco e remover, são exemplos de tarefas rotineiras que podem ser automatizadas por meio de um script.

Com base neste cenário, o objetivo deste artigo é proporcionar ao usuário que já possui certo domínio na utilização do Linux, uma visão sobre o ambiente shell e a familiarização com as características básicas (estrutura, variáveis, parâmetros) para a criação de scripts utilizando o shell Bash.

Definição inicial de Shell

De uma forma geral, shell é o nome comum para um conjunto de programas que tem o propósito de fornecer uma interface

interativa com o usuário, desempenhando entre outras, as seguintes funções:

- Ler a linha de comando do usuário;
- Separar o comando e parâmetros passados;
- Executar o comando (interno ou não);
- Esperar que o processo seja finalizado;
- Retornar o controle para o usuário.

Tipos de shell

O shell, como já informado, é o programa que permite a interação do usuário com o sistema. No início, o sh era o shell padrão utilizado para criação de scripts e interpretação de comandos, porém, atualmente existem diversos tipos de shell, cada um com seus próprios recursos, capacidades e limitações, sendo que em quase todas as distribuições do Linux, o bash é o shell padrão. A seguir são apresentadas as descrições dos principais tipos de shells.

sh

Bourne shell, ou simplesmente sh, foi o shell padrão do Unix Versão 7, que substituiu o Thompson shell, cujo arquivo executável tinha o mesmo nome, sh. Desenvolvido por Stephen Bourne, dos laboratórios AT&T, foi lançado em 1977 junto com o Unix Versão 7 distribuído para as universidades. Shell com maior portabilidade para criação de scripts, introduziu uma sintaxe compacta para canalização e redirecionamento (`|`, `<>` e `>>`), suporte para comandos sequenciais (utilizando o `;`) e recursos como histórico e edição de comandos, alias, entre outros.

csh/tcsh

O C shell (csh) foi desenvolvido para sistemas UNIX Berkeley Software Distribution (BSD) por Bill Joe, com o objetivo de criar uma linguagem de script que fosse semelhante à linguagem C. Um recurso útil introduzido por Bill Joe no csh foi o histórico de comandos, que permitia que o usuário revisasse e selecionasse facilmente comandos anteriores para serem executados. Após cinco anos, Ken Greer, da Universidade Carnegie Mellon, criou o tcsh, compatível com scripts C Shell e com recursos como auto completar melhorado em relação ao csh, correção ortográfica e edição de linha de comando.

ksh

O Korn Shell (ksh) foi projetado por David Korn na mesma época que o tcsh, sendo compatível com o Bourne Shell. Inclui recursos como gerenciamento do histórico de comandos, uso de alias, auto completar, controle de jobs e interação com outras linguagens como awk, perl e tcl. Em função do seu licenciamento inicial, era software proprietário até 2000, quando foi lançado como software livre (sob a licença Common Public License).

bash

Bourne-Again Shell (trocadilho com Stephen Bourne, autor do Unix Shell sh), ou Bash, é o interpretador de comandos para sistemas GNU. Desenvolvido por Brian Fox, é padrão na maioria das

distribuições Linux atuais, como Red Hat, Suse, Debian e outras, sendo amplamente compatível com sh, além de incorporar características úteis do ksh e do csh, oferecendo melhorias funcionais sobre o sh tanto para uso interativo como para programação. Atualmente, funciona em quase todas as versões do Unix e em alguns outros sistemas operacionais, como Linux, Windows, Darwin e outros. Dentre as características, pode-se citar o histórico de comandos, controle de tarefas agendadas, funções de shell, aliases, operações aritméticas, personalização, suporte para expressões regulares (semelhante ao Perl), arrays associativos e muitas outras. Este shell será o objeto de estudo deste artigo em um ambiente GNU/Linux, modo texto.

Arquivos de configuração do Shell

Toda sessão no Linux (seja modo gráfico ou texto) se inicia com a solicitação de um nome de login e senha, e uma vez que essas informações estejam corretas, um shell é invocado e fará a leitura de alguns arquivos de configuração do ambiente antes de entregar o controle para que o usuário possa interagir com o sistema por meio de comandos.

O bash é o shell padrão nas distribuições Linux, e é utilizado caso o arquivo `/etc/passwd` não especifique outro shell para o usuário, ou indique o caminho de um script a ser executado após o usuário fazer login no sistema. A **Listagem 1** apresenta o conteúdo do arquivo `/etc/shells` de um sistema Debian GNU/Linux Squeeze, que contém uma lista de interpretadores de comando (shells) válidos no sistema, lembrando que este conteúdo pode variar entre as distribuições.

Listagem 1. Conteúdo do arquivo `/etc/shells`.

```
# cat /etc/shells
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
```

Existem duas maneiras de se verificar qual Shell está sendo utilizado no ambiente:

- 1) consultando a variável de ambiente `SHELL`; e,
- 2) verificando o último parâmetro definido para o usuário no arquivo `/etc/passwd`, conforme apresentado na **Listagem 2**.

Os arquivos de configuração do shell definem, entre outras coisas, os aliases, tipo de prompt e variáveis de ambiente, e podem ser classificados como globais, influenciando na configuração do ambiente de todos os usuários ou locais, atuando somente na

sessão do usuário. A leitura dos arquivos de configuração obedece à seguinte ordem:

1. Quando um shell de login interativo (isto é, que requer autenticação) é iniciado, o arquivo `/etc/profile`, caso exista, é executado. Esse arquivo contém configurações globais que são aplicadas a todos os usuários do sistema;
2. Em seguida, o shell procura pelos seguintes arquivos: `~/.bash_profile`, `~/.bash_login` ou `~/.profile` no diretório home do usuário, executando o primeiro que estiver disponível com permissão de leitura. Caso algum dos arquivos exista, mas não possa ser lido, será reportado um erro para o usuário;
3. Durante a sessão, caso seja invocado um novo shell através da ação do usuário executando o comando `xterm` para abrir uma nova janela de terminal, o arquivo `~/.bashrc` é lido e executado. Esse arquivo contém as configurações de personalização do ambiente, tais como definição de alias, cores do prompt, entre outras;
4. Quando o shell for finalizado (usuário efetuar o logout), será lido e executado o arquivo `~/.bash_logout`, caso exista.

Listagem 2. Consultando o Shell utilizando no ambiente.

Utilizando o comando `echo`:

```
# echo $SHELL
/bin/bash
```

Trecho do conteúdo do arquivo `/etc/passwd`:

```
# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:2:2:bin:/bin:/bin/sh
ivani:x:1000:1000:lvani,,/home/ivani:/bin/bash
pgoncales:x:1009:1004:Pathiene,,/home/pgoncales:/bin/bash
```

Uma vez que os arquivos de configuração do ambiente foram lidos, o shell entrega o controle para o usuário e está pronto para ser utilizado para a execução de tarefas de administração de sistemas. Estas tarefas podem variar desde a criação de usuários e grupos, leitura de logs, até a programação em shell e a criação de scripts.

Shell Script

Como já explanado neste artigo, o shell é um interpretador de comandos, sendo uma interface para comunicação entre o usuário e o kernel. Quando o usuário digita um comando, este é analisado pelo shell, que verifica as permissões, resolve redirecionamentos, substitui variáveis e metacaracteres, caso existam, e finalmente solicita ao kernel a execução das ações correspondentes ao comando em questão. Imaginando que um usuário necessite executar sempre esse mesmo conjunto de comandos, seria ideal automatizar a rotina por meio da utilização de um script. Portanto, quando existe a necessidade de executar a mesma tarefa diariamente, como criar um backup do diretório `/etc` em vez de digitar os comandos repetidamente, basta colocá-los na sequência correta dentro de um arquivo texto e transformá-lo em um roteiro, que será lido, interpretado

e executado pelo shell. Esse roteiro que foi descrito, comumente chamado de shell script, é frequentemente utilizado para tarefas de administração de sistemas ou para combinar programas existentes para concluir um trabalho específico.

Aplicações do uso do Shell script

É possível utilizar o shell de forma interativa e não-interativa. Quando o usuário realiza uma operação simples, como a execução de algum comando como *ls*, *cat* ou *cd*, o shell está sendo utilizado interativamente. Porém, essa forma de uso pode ser cansativa, se as operações são repetidas continuamente, sendo mais prático colocar esses comandos em um arquivo para reaproveitamento, isto é, criar um script.

O uso automatizado (não-interativo) do Shell Script (utilizando arquivos com comandos) deve ser feito quando for necessário criar uma rotina a ser executada por todos os usuários sem necessitar digitá-la comando a comando; procedimentos complexos usando muitas linhas de comando; execução de uma tarefa planejada para ocorrer em uma determinada data; interação de informações de vários sistemas existentes; tarefas rotineiras; etc.

Criando um script simples

Antes de partir para a criação de scripts elaborados, vejamos como criar um script simples, com a finalidade de contar quantos usuários estão conectados no sistema. Para criar este script, pode ser utilizado o comando *who*, que exibe os usuários conectados linha a linha, conforme mostra a saída do comando na **Listagem 3**.

Listagem 3. Verificação de usuários conectados no sistema.

```
# who
ivani    tty1    2013-02-27 02:42
root     pts/    2013-02-27 02:42 (192.168.1.103)
pgoncal  pts/    2013-02-27 02:44 (192.168.1.103)
```

No exemplo, somente três usuários estão conectados, mas em um sistema multiusuário, essa listagem pode ser maior e a rolagem da tela poderia não permitir contar todos os usuários de uma forma manual. Nesse caso, o usuário pode optar por automatizar essa tarefa, mas antes é necessário contar o número de usuários. Isso pode ser feito combinando o comando *who* com o comando *wc* (*word count*), que conta linhas, palavras e caracteres, conforme apresentado a seguir:

```
# who | wc -l
3
```

A opção *-l* (line) do comando *wc*, exibe a quantidade de linhas, e o símbolo *|* (pipe) cria um canal entre dois comandos: a saída do comando *who* torna-se a entrada do comando *wc*, que imprime a quantidade de usuários conectados.

Sabendo o número de usuários, pode-se automatizar a tarefa, bastando para isso escrever os comandos em um arquivo de

texto normal e torná-lo executável utilizando o comando *chmod*, conforme o exemplo na **Listagem 4**. O comando *cat* é utilizado para escrever em arquivo texto, exibir seu conteúdo ou ainda concatenar arquivos. Vale lembrar que este comando não substitui o editor de textos, uma vez que sua função nesse exemplo é apresentar mais uma forma de realizar a tarefa de criação e edição de arquivos texto no terminal.

Listagem 4. Automatizando a contagem de usuários.

Cria o arquivo e copia o conteúdo do arquivo utilizando o comando *cat*:
cat > conta_users

Conteúdo do script
who | wc -l

CTRL-D para salvar e finalizar a edição do script
^D
Torna o arquivo executável com a opção *+x* do comando *chmod*:
chmod +x conta_users

Executa o script no diretório onde foi criado, sendo que a saída deve retornar o número de usuários:
./conta_users
3



Não perca tempo reinventando a roda!

COBREBEMX

Componente completo para sua Cobrança por Boleto Bancário e Débito em Conta Corrente

Mais de 40 exemplos em diversas linguagens de programação

Geração e leitura de arquivos (remessa e retorno) nos padrões FEBRABAN e CNAB

Testes e Downloads gratuitos em nosso site

ACESSE E CONHEÇA O COMPONENTE EM:
WWW.COBBREBEM.COM

Estrutura de um script

Neste tópico, será apresentada a estrutura de um script, isto é, comandos colocados em um arquivo para serem executados posteriormente. Os scripts foram elaborados em um servidor com o sistema operacional Debian GNU/Linux 6.0, e o editor de textos utilizado foi o *vim*, porém o usuário pode utilizar qualquer editor sobre o qual tenha conhecimento.

Inicialmente, para deixar o estudo de shell script organizado, é interessante criar um diretório para armazenar os scripts que serão criados, conforme o exemplo apresentado na **Listagem 5**.

Listagem 5. Criando diretório para armazenar os scripts criados.

```
Entra no diretório home do usuário
$ cd /home/aluno

Cria o diretório scripts:
$ mkdir scripts

Verifica o local onde se encontra (pwd) e lista o conteúdo do diretório:
$ pwd
/home/aluno
$ ls -l
drwxr-xr-x 2 aluno aluno 4096 Feb 27 04:10 scripts

Entra no diretório onde serão armazenados o scripts:
$ cd scripts/
$ pwd
/home/aluno/scripts
```

No exemplo, o diretório *script* foi criado dentro do home do usuário aluno. Após a criação, listou-se o conteúdo do home e em seguida foi executado o comando *cd* para entrar no diretório recém-criado. A partir de agora, os scripts apresentados neste artigo serão criados utilizando o usuário aluno dentro do diretório */home/aluno/scripts*. Vale lembrar que este é um usuário comum; caso seja necessária a utilização do usuário root, o fato será mencionado.

No primeiro contato com shell script (vide **Listagem 4**), foi apresentado um exemplo de script simples, porém não foi utilizado qualquer tipo de estrutura convencional em sua criação (definição de shell padrão, comentários, etc.). A **Listagem 6** apresenta um script que exibe a data e a hora, lista o conteúdo do diretório e exibe o espaço livre em disco utilizando práticas comuns na programação em shell.

Listagem 6. Exemplo de estrutura de shell script.

```
$ vim shellscrip.sh
#!/bin/bash
#Script de teste
date
ls -l
df -h
```

Ainda que o Shell Script seja escrito em um editor de textos, o arquivo geralmente segue uma extensão padrão, que é a extensão *.sh* (shell), que pode ser definida ao criar o arquivo. Um fator importante que deve ser considerado na hora de definir o nome do script é evitar o uso de letras maiúsculas, acentuação

e espaçamento. Sendo assim, o ideal é utilizar nomes com letras minúsculas, que podem ser separados por - (hífen) ou _ (underline) se necessário, como *shell-script.sh* ou *shell_script.sh*, conforme o exemplo a seguir:

```
$ vim shellscrip.sh
```

Este comando utilizou o editor de textos *vim* para criar o arquivo *shellscrip.sh*. É neste arquivo que serão inseridos os comandos do script.

Uma vez que existem diversos shells, é necessário saber o shell padrão utilizado pelo sistema operacional, pois a primeira linha do script indica qual shell será utilizado para interpretar os comandos. Para saber qual o shell padrão, basta executar:

```
$ echo $SHELL
```

Feito isso, adicione os comandos apresentados na **Listagem 7** no arquivo *shellscrip.sh*.

Listagem 7. Conteúdo do script shellscrip.sh.

```
#!/bin/bash
#Script de teste
date
ls -l
df -h
```

Como já informado, a primeira linha é utilizada por padrão para definir o shell. A segunda linha é apenas um comentário. O sinal “#” serve para permitir a adição de comentários dentro do script. Esses comentários não são impressos na tela quando o script é executado.

Da terceira até a quinta linha são executados três comandos: *date*, *ls -l* e *df -h*. O comando *date* sem parâmetros imprime a data e a hora corrente, o comando *ls* seguido do parâmetro *-l* (*list*) lista o conteúdo do diretório corrente, e o comando *df* seguido do parâmetro *-h* (*human*) mostra o espaço que o disco possui separado pelas partições.

Depois que os comandos foram inseridos no script é necessário salvar as informações, fechar o editor e dar a permissão de execução ao arquivo:

```
$ chmod +x shellscrip.sh
```

O comando *chmod* seguido do parâmetro *+x* é o que definirá a permissão de execução ao arquivo *shellscrip.sh*. Agora que o script já está pronto para ser executado, pode-se utilizar duas maneiras para iniciá-lo, a saber:

```
$ sh shellscrip.sh
```

ou

```
$ ./shellscrip.sh
```

O resultado da execução do script será o mesmo nas duas opções. A diferença é que, ao empregar o comando *sh*, é solicitado ao shell que execute o script mesmo que você esteja em um diretório diferente do local original do script. Já o *./* executa o script pelo binário; portanto é necessário que ao utilizar esse método, você esteja no mesmo diretório em que o script foi criado.

Um ponto curioso é que, ao usar o comando *sh* para executar um script, não é obrigatório ter a permissão de execução. Neste caso, independentemente das opções que você possui para executar um script, os resultados apresentados serão os mesmos.

Quando se está criando um script, é interessante imprimir mensagens na tela para apresentar o que será executado. O comando que imprime as mensagens é o *echo*. Com este comando também é possível verificar o valor das variáveis do sistema. Por exemplo, o comando *echo \$BASH* apresenta o valor da variável *BASH*. A mesma informação pode ser obtida com a variável *SHELL*. Ambos os resultados apresentam o shell padrão que está sendo utilizado para executar os comandos.

Nota-se que para visualizar os valores das variáveis é necessário utilizar o sinal de '\$' antes do nome da variável, caso contrário, o comando *echo* apenas imprimirá o que foi escrito no shell, conforme apresentado na **Figura 2**.

```
pgoncales@machine:~$ echo $SHELL
/bin/bash
pgoncales@machine:~$ echo $BASH
/bin/bash
pgoncales@machine:~$ echo $SHELL
SHELL
pgoncales@machine:~$ echo $BASH
BASH
```

Figura 2. Visualização do conteúdo da variável

Quando o script precisar de alterações, basta editar o arquivo novamente e inserir ou alterar os comandos necessários. As alterações apresentadas na **Listagem 8** farão com que sejam impressas mensagens na tela quando o script for executado. Para isto, empregaremos o comando *echo*.

Listagem 8. Utilizando echo para exibir mensagens na tela.

```
$ vim shellsript.sh
#!/bin/bash
#Script de teste
echo "Data"
sleep 1
date
echo "Listagem do diretório corrente"
sleep 2
ls -l
echo "Espaço em Disco"
sleep 3
df -h
```

Outro comando adicionado é o *sleep*, que faz uma breve pausa em segundos. No script, o comando *sleep 1* fará uma pausa de 1 segundo e o comando *sleep 3* fará uma pausa de três segundos.

Manipulando variáveis

Assim como todas as linguagens de programação, o shell também permite o uso de variáveis na construção de scripts. Para facilitar a compreensão de como utilizá-las, a **Listagem 9** demonstra um exemplo.

Listagem 9. Trabalhando com variáveis.

```
#!/bin/bash
#Script de teste
echo -e "Executando script...\n Permite que os comandos a seguir sejam
executados? [s/n]"
read RESP
test "$RESP" = "n" && exit
echo "Data"
sleep 1
date
echo "Listagem do diretório corrente"
sleep 2
ls -l
echo "Espaço em Disco"
sleep 3
df -h
```

Na primeira linha em destaque na **Listagem 9** é possível notar o comando *echo*, utilizado para exibir mensagens na tela. Nesse ponto, quando o script for executado, a seguinte mensagem será exibida:

Executando o script...

Permite que os comandos a seguir sejam executados? [s/n]

Note que o comando *echo* está acompanhado da opção *-e* e embora a mensagem para o usuário esteja em apenas uma linha, observe que ao executar o script, a sentença foi quebrada em duas linhas. Esta opção permite que o *echo* interprete caracteres especiais que não são impressos, como por exemplo, a quebra de linha (*\n*) ou ainda tabulações horizontais (*\t*) ou verticais (*\v*).

Continuando com a análise das linhas em destaque na **Listagem 9**, após exibir a mensagem na tela, o script aguarda por uma ação do usuário que definirá, neste caso, se a execução do script deve prosseguir ou não. O comando *read* faz a leitura do que foi digitado pelo usuário e armazena este resultado em uma variável chamada *RESP*, conforme a linha a seguir:

read RESP

O valor armazenado na variável *\$RESP* será analisado pela seguinte condição: se a resposta digitada pelo usuário for igual a *n*, então o script será finalizado. Se a resposta do usuário for *s*, então o script seguirá com a execução. Esta condição está representada na linha:

test "\$RESP" = "n" && exit

O comando *test* é um condicional (assim como o *if*, *case*, entre outros) e o operador lógico “&&” significa que o segundo comando só será executado se o primeiro comando der uma resposta positiva.

A Figura 3 apresenta a execução do script.

```
pgoncales@machine:~$ ./shellscript.sh
Executando script...
Permite que os comandos a seguir sejam executados? [s/n]
n
pgoncales@machine:~$ ./shellscript.sh
Executando script...
Permite que os comandos a seguir sejam executados? [s/n]
s
"Data"
Tue Feb 26 15:20:40 BRT 2013
"Listagem do diretório corrente"
```

Figura 3. Exemplo de script com interação do usuário

Como pode ser observado, o script foi executado duas vezes para apresentar os diferentes resultados que podem ser obtidos. Na primeira execução, o usuário digitou a opção *n* e o script foi interrompido. Já na segunda, o usuário digitou *s* e o script executou todos os comandos que estavam configurados.

Trabalhando com parâmetros

Em algumas situações, os scripts precisam de parâmetros para que possam ser executados. Por exemplo, vamos supor que você crie um script que faça a atualização de um arquivo. Como o nome do arquivo pode ser diferente a cada atualização, é necessário passar através de um parâmetro o arquivo novo que o script terá que utilizar.

Ao passar o parâmetro para um script, este valor é armazenado em uma variável de modo que dentro do script seja possível trabalhar com esta variável da maneira que você preferir.

A passagem de parâmetros é feita no ato da chamada para a execução do script. Para auxiliar nessa tarefa, são utilizadas variáveis pré-definidas que são tratadas de maneira especial pelo shell. Esse tipo de variável pode ser utilizado para saber, por exemplo, quantos parâmetros foram passados para o script em linha de comando, nome do script, número do processo, etc. A Tabela 1 apresenta uma lista dos principais parâmetros e sua descrição.

A seguir, serão apresentados alguns exemplos de scripts que utilizam passagem de parâmetros e código de retorno.

Parâmetro	Descrição
\$0	Nome do script que está sendo executado
\$1 - \$9	Parâmetros passados na linha de comando
\$#	Número total de parâmetros na linha de comando
\$\$	Número PID (Process ID) do processo
\$_	Código de retorno do último comando executado

Tabela 1. Lista dos principais parâmetros especiais do shell

\$1 - \$9 – Passagem de parâmetros

O script demonstrado na Listagem 10 tem como objetivo apresentar as funções mais básicas quando se trabalha com parâmetros em shell script, que são: o envio de um parâmetro, o armazenamento desta informação e como ela é tratada.

Listagem 10. Exemplo de script que utiliza passagem de parâmetro.

```
$ vim parametros.sh

1 #!/bin/bash
2 #Script que faz a leitura de parâmetros digitados pelo usuário
3 echo "Os parâmetros digitados são:"
4 echo $1
5 echo $2
6 echo $3
```

Ao executar o script, os parâmetros devem ser passados juntamente com o nome do script, conforme o exemplo a seguir:

```
./parametros.sh inframagazine shell script
```

A Figura 4 apresenta o resultado da execução do script.

```
pgoncales@machine:~$ ./parametros.sh inframagazine shell script
Os parâmetros digitados são:
inframagazine
shell
script
```

Figura 4. Execução de script utilizando passagem de parâmetros

Na Listagem 10, as linhas foram numeradas para facilitar a explicação de todo o script. A primeira linha define qual o *bash* padrão que está sendo utilizado para executar e interpretar os comandos no script. A segunda linha é apenas um comentário para representar o que o script executará. O comando *echo* imprimirá a mensagem na terceira linha e, por fim, nas linhas 4, 5 e 6, o comando *echo* fará a leitura de cada variável, em que cada uma delas recebeu o valor do parâmetro digitado no início do script.

\$# – Número total de parâmetros

O uso deste parâmetro fornece o número total de argumentos passados para o shell script na linha de comando. É utilizado em estruturas de repetição (o que será abordado mais adiante) para processamento de opções e argumentos. O script da Listagem 10 apresentou os parâmetros digitados pelo usuário, e com uma pequena alteração, o mesmo script mostrará a quantidade e os parâmetros passados na linha de comando, conforme demonstrado a seguir:

```
1 #!/bin/bash
2 # Script que exibe a quantidade e parâmetros passados
3 echo -e "Os $# parâmetros digitados são: \n $1 \n $2 \n $3"
```

No exemplo, o script da Listagem 10 foi reduzido a apenas uma linha de execução, onde serão exibidos a quantidade e os parâ-

tros passados pelo usuário na linha de comando. Sendo assim, o resultado obtido ao executar este script será:

```
$ ./parametros.sh inframagazine shell script
Os 3 parâmetros digitados são:
inframagazine
shell
script
```

\$? – Código de retorno

Todo comando executado no shell retorna um valor depois que é encerrado. Este valor é definido como o estado de saída ou código de retorno e pode auxiliar na análise para saber se um comando foi executado corretamente. Se o código de retorno for 0 (zero), significa que o comando foi executado com sucesso. Por exemplo, quando o usuário deseja copiar um arquivo, ele utiliza o comando *cp*. Quando a execução deste comando é encerrada, o shell armazena o estado de saída do *cp* na variável *\$?*. Assim, para saber se o comando foi executado corretamente, basta verificar se o valor da variável *\$?* é igual a zero. A **Listagem 11** apresenta alguns exemplos relacionados.

Listagem 11. Exemplos de análise do código de retorno ao executar cópia de arquivos.

Cópia de um arquivo existente:
cp -v /etc/hosts /tmp
'/etc/hosts' -> '/tmp/hosts'

Código de retorno indica sucesso:
echo \$?
0

Cópia de um arquivo inexistente com mensagem de erro do comando cp:
cp -v /etc/host /tmp/
cp: cannot stat '/etc/host': No such file or directory

Código de retorno indica falha:
echo \$?
1

Expressões aritméticas

As expressões aritméticas podem ser utilizadas no shell script para efetuar operações matemáticas básicas como adição, subtração, multiplicação e divisão, além de diversas outras opções. Quem auxilia na execução dessas expressões é o comando *expr* (*evaluate expressions*), conforme apresentado no comando a seguir:

```
$ expr 2 + 2
4
```

Este comando trata os argumentos que são apresentados a ele e faz cálculos ou diversas outras ações que podem ser utilizadas, conforme pode ser visto na **Tabela 2**. Vale ressaltar que este comando trabalha apenas com números, portanto, na tentativa de realizar junções como *a + a*, será retornado um erro, conforme demonstrado na **Figura 5**.

Argumento	Definição
ARG1 < ARG2	Compara o ARG1 se é menor que o ARG2
ARG1 <= ARG2	Compara o ARG1 se é menor ou igual que o ARG2
ARG1 = ARG2	Compara o ARG1 se é igual ao ARG2
ARG1 != ARG2	Compara o ARG1 se é diferente do ARG2
ARG1 >= ARG2	Compara o ARG1 se é maior ou igual que o ARG2
ARG1 > ARG2	Compara o ARG1 se é maior que o ARG2
ARG1 + ARG2	Efetua a soma dos dois argumentos
ARG1 - ARG2	Efetua a subtração dos dois argumentos
ARG1 * ARG2	Efetua a multiplicação dos dois argumentos
ARG1 / ARG2	Efetua a divisão dos dois argumentos
ARG1 % ARG2	Apresenta o resto da divisão entre ARG1 e ARG2

Tabela 2. Argumentos do comando *expr*

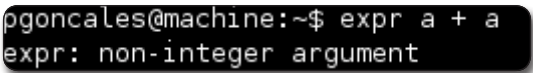


Figura 5. Erro produzido ao utilizar comando *expr* com letras

Na **Listagem 12** é demonstrado um script que irá executar a soma entre dois números utilizando o comando *expr*, que pegará os valores para efetuar os cálculos por meio de parâmetros.

Listagem 12. Exemplo de script com a operação aritmética soma.

```
$vim soma.sh
1 #!/bin/bash
2 #Script que faz a soma entre dois números que devem ser passados por parâmetro
3 echo "Os números digitados são:"
4 echo $1
5 echo $2
6 echo "O resultado da soma destes dois números é:"
7 soma=$(expr $1 + $2)
8 echo $soma
```

Na linha 1 está definido o shell padrão e na linha 2 foi definido um breve comentário sobre o objetivo do script. As linhas 3 a 6 imprimem mensagens na tela com o comando *echo*. Para indicar que uma variável receberá um valor que será o resultado de um comando, é necessário utilizar a estrutura *\$(comando)*. Na linha 7 é armazenado um valor na variável “soma”. O valor que esta variável recebe vem do cálculo realizado pelo comando *expr*. A linha 8 apenas imprime o resultado da soma. Por fim, aplica-se a permissão de execução e basta rodar o script conforme apresentado na **Figura 6**.

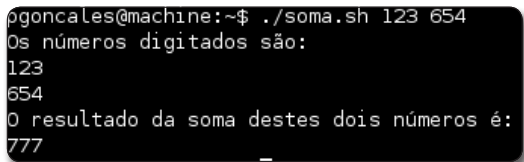


Figura 6. Execução do script de operação aritmética

Controle de fluxo

Uma das principais vantagens na utilização de um shell script é a possibilidade de agilizar a execução de tarefas de alta demanda, tais como manipulação de listas de dados, criação de usuários e grupos, backups, definição de quotas em disco, entre outras. Este tópico tratará de estruturas de controle de fluxo que dão a inteligência para o script decidir o que fazer a partir da utilização de controles condicionais (estruturas de decisão) ou loops (estruturas de repetição).

Estruturas Condicionais

Estruturas condicionais são recursos que possibilitam a tomada de decisão a partir de um teste que verifica se a condição para execução de determinado trecho de código é verdadeira ou falsa. Dependendo da análise, tal trecho poderá ou não ser executado.

if

Em toda linguagem de programação existe um tipo de estrutura para tomada de decisões. O *if*, um comando condicional popular, executa um bloco de comandos caso uma determinada expressão seja verdadeira (*if*); senão (*else*) abre um bloco de comandos para serem executados caso a expressão em teste seja falsa. No caso da necessidade de criação de ifs aninhados, isto é, um *if* dentro de outro, pode-se utilizar o comando *elif* (junção de *else* e *if*), que substitui a cadeia de *else/if*, melhorando a legibilidade do código e otimizando a execução do script. A **Listagem 13** apresenta a sintaxe para utilização do *if* e *elif*.

Listagem 13. Sintaxe da estrutura condicional *if* e *elif*.

```
if condição
then
    Bloco de comandos 1
elif condição
then
    Bloco de comandos 2
else
    Bloco de comandos 3
fi
```

Na sintaxe apresentada, a primeira condição é testada. Se retornar verdadeiro, o bloco de comandos 1 será executado, senão, o fluxo de execução passará para a próxima condição, que é o *elif*. Dentro do *elif* existe uma segunda condição, e se esta for verdadeira, então o bloco de comandos 2 será executado. Se nenhuma das condições anteriores for verdadeira, a execução passará para o bloco de comandos 3.

A seguir, são apresentados dois exemplos de uso do *if* combinados com recursos já vistos no artigo, como interação com o usuário por meio do comando *read*, substituição de variáveis e utilização de parâmetros. As linhas do script foram numeradas para facilitar a compreensão.

O exemplo da **Listagem 14** testa se a string digitada pelo usuário é igual ao conteúdo da variável de ambiente *\$USER*, comparando se o nome de usuário digitado é o mesmo que está conectado no momento da execução do programa.

Listagem 14. Exemplo de uso do *if*.

```
$ vim usuario.sh
1 #!/bin/bash
2 echo "Teste com o comando if"
3 echo "Entre com seu nome de usuário:"
4 read NOME
5 if [ $NOME = "$USER" ]
6 then
7   echo "Bem vindo $USER"
8 else
9   echo "Usuário não reconhecido"
10 fi
```

Considerando que você está no diretório *scripts* criado no início do artigo, após editar o arquivo *usuario.sh*, é preciso salvar e dar permissão de execução para o mesmo com o comando *chmod*:

```
$ chmod +x usuario.sh
```

Na análise do código, verifica-se que o comando *echo* exibe mensagens na tela e solicita que o nome do usuário seja digitado. A entrada é recebida pelo comando *read* e armazenada em uma variável chamada *NOME*, conforme o trecho do código a seguir:

```
2 echo "Teste com o comando if"
3 echo "Entre com seu nome de usuário:"
4 read NOME
```

Em seguida, o *if* testa se o nome que o usuário digitou é o mesmo que está armazenado na variável *\$USER*, que é responsável por guardar o nome do usuário conectado na máquina no momento.

Cabe dizer aqui que o teste realizado pelo *if* é feito por meio do comando *test*, mencionado anteriormente neste artigo, e que tem por função processar a expressão condicional, retornando um código 0 ou 1. No exemplo, o comando *test* é representado por colchetes (*[]*) e a expressão a ser testada deve ser separada por espaços, conforme o código apresentado a seguir:

```
5 if [ $NOME = "$USER" ]
6 then
7   echo "Bem vindo $USER"
```

Caso o conteúdo da variável *\$NOME* seja igual ao da variável *\$USER*, a mensagem de boas vindas com este nome será exibida na tela, conforme a execução do script no bloco *if*, apresentado na **Listagem 15**.

Se o conteúdo da variável *\$NOME* for diferente do conteúdo armazenado na variável *\$USER*, será exibida a mensagem "Usuário não reconhecido" na tela e o script será finalizado com o comando *fi*, como indica o trecho do código:

```
8 else
9   echo "Usuário não reconhecido"
10 fi
```

A **Listagem 16** apresenta a execução do script no bloco *else*.

A seguir, a **Listagem 17** apresenta o uso do *if* e do *elif* em um exemplo em que é solicitado ao usuário que informe dois números inteiros. Após a comparação, é exibida uma mensagem na tela, informando se os números são iguais ou se um é menor ou maior que o outro.

Listagem 15. Execução do script usuario.sh no bloco if.

Execução do script em linha de comando:
aluno@lab:~/scripts\$./usuario.sh
Teste com o comando if
Entre com seu nome de usuário:
aluno
Bem vindo aluno

Comando para verificar o conteúdo da variável \$USER:
aluno@lab:~/scripts\$ echo \$USER
aluno

Listagem 16. Exemplo do script usuario.sh no bloco else.

aluno@lab:~/scripts\$./usuario.sh
Teste com o comando if
Entre com seu nome de usuário:
asmith
Usuário não reconhecido

Listagem 17. Exemplo de uso do if e do elif.

\$ vim compara_numeros.sh
01 #!/bin/bash
02 if [\$# -ne 2]
03 then
04 echo "Insira dois números para comparação"
05 echo "Uso: ./compara_numeros.sh num1 num2"
06 exit
07 elif [\$1 -eq \$2]
08 then
09 echo "\$1 e igual a \$2"
10 elif [\$1 -lt \$2]
11 then
12 echo "\$1 e menor que \$2"
13 else
14 echo "\$1 e maior que \$2"
15 fi

Após editar o script, deve-se salvar e dar permissão para execução utilizando o comando *chmod*:

```
$ chmod +xcompara_numeros.sh
```

Este script tem por objetivo comparar dois valores, sendo que, na análise do código, o *if* testa inicialmente se o usuário passou a quantidade de parâmetros necessária para realizar a comparação entre dois valores utilizando a variável especial *\$#*. Caso não tenha os parâmetros necessários, é exibida uma mensagem de como executar o programa e o script é encerrado, conforme apresentado no trecho do código a seguir:

```
2 if [ $# -ne 2 ]  
3 then  
4 echo "Insira dois números para comparação"  
5 echo "Uso: ./compara_numeros.sh num1 num2"  
6 exit
```

Nesse script, para efeito de comparação numérica, foi utilizado o operador *NotEqual* (*-ne*), isto é, se a quantidade de parâmetros que o usuário passou não for igual a 2, a mensagem de alerta é exibida e o script é encerrado. A **Tabela 3** apresenta os operadores que podem ser utilizados para comparação numérica.

Operador	Descrição
-lt	LessThan – é menor que
-gt	GreaterThan – é maior que
-le	LessEqual – é menor ou igual
-ge	GreaterEqual – é maior ou igual
-eq	EQual – é igual
-ne	NotEqual – é diferente de

Tabela 3. Operadores utilizados em testes de comparação numérica

Prosseguindo com a análise do código do script *compara.sh*, se o usuário passou a quantidade de parâmetros correta, o bloco *elif* é executado.

Nesse bloco de código, os números digitados são comparados por meio dos operadores *-eq* (EQual) e *-lt* (LessThan), respectivamente, exibindo uma mensagem ao usuário. Por fim, se nenhum dos blocos mencionados corresponderem aos números que o usuário digitou, o bloco *else* será executado exibindo a mensagem de que o primeiro número digitado é maior que o segundo, conforme mostra o trecho do código:

```
13 else  
14 echo "$1 e maior que $2"  
15 fi
```

A **Listagem 18** apresenta o script *compara_numeros.sh* em execução.

Listagem 18. Script compara_números.sh em execução.

Script executado sem os parâmetros:
[aluno@lab: ~/scripts]\$./compara_numeros.sh
Insira dois números para comparação
Uso: ./compara_numeros.sh num1 num2

Script executando comparações de números:
[aluno@lab: ~/scripts]\$./compara_numeros.sh 7 7
7 e igual a 7

[aluno@lab: ~/scripts]\$./compara_numeros.sh 5 9
5 e menor que 9

[aluno@lab: ~/scripts]\$./compara_numeros.sh 2 17
2 e menor que 17

[aluno@lab: ~/scripts]\$./compara_numeros.sh 17 2
17 e maior que 2

case

O comando *case* executa um bloco de código de acordo com o valor de uma variável, sendo recomendado seu uso quando

a quantidade de comandos condicionais (*if*) sucessivos for maior que três. Essa prática permite uma execução ágil, aumenta a legibilidade do código, bem como diminui o seu tamanho. Normalmente o *case* é utilizado para construção de interfaces de menu que tomarão alguma decisão baseada em várias opções de entrada do usuário. A **Listagem 19** apresenta a sintaxe do comando *case*.

Listagem 19. Sintaxe estrutura condicional *case*.

```
case <valor> in
  opcao1)
bloco de comandos
;;
  opcao2)
bloco de comandos
;;
*)
  operação padrão
esac
```

Na sintaxe apresentada, *<valor>* é comparado de forma seletiva a cada um dos padrões (*opcao1* e *opcao2*) até que um deles seja satisfeito, quando então passará a executar o bloco de comandos subsequentes até encontrar dois pontos-e-vírgulas sucessivos (*;;*) que indicam o fim do bloco de comandos.

Caso o valor informado não satisfaça nenhum dos padrões, será executado o bloco de comandos da entrada **)* e o *case* será finalizado com o comando *esac*.

A **Listagem 20** demonstra um exemplo de utilização da estrutura condicional *case* para uma calculadora de números inteiros, em que são combinados recursos como exibição de mensagem na tela, teste de parâmetros e operações aritméticas no shell.

O script apresentado tem por objetivo executar operações aritméticas por meio de valores passados via linha de comando e o tipo de operação que deseja realizar. O comando *case* analisa o conteúdo das variáveis *\$1* e *\$2* (primeiro e segundo parâmetros passado para o programa, respectivamente) e compara se o valor da variável *\$3* coincide com as opções que possui. Se a variável *\$3*, por exemplo, for *soma*, o bloco de comandos referente à soma será executado e assim por diante.

A opção **)* é padrão no *case*. Deste modo, caso o parâmetro passado não seja igual a nenhuma das outras opções anteriores, esse bloco de comandos será executado.

Estruturas de Repetição

Todo programa tem uma entrada de dados, processamento e saída, sendo que muitas vezes o processamento é feito em ciclos, que normalmente são chamados de laços. As estruturas de repetição ou laços são utilizadas para repetir um bloco de comandos por uma quantidade de vezes determinada por um limite que é definido no script na própria estrutura de repetição.

for

O laço *for* utiliza uma lista na qual os valores são atribuídos um a um a uma variável de controle, sendo que a cada atribuição de

valor à variável de controle o bloco de comandos *do* é executado. A **Listagem 21** apresenta a sintaxe desse comando.

Na sintaxe apresentada, a cada repetição o *for* associa um item da lista *<list>* à variável *<var>* e executa os comandos até que toda a lista seja percorrida. A **Listagem 22** apresenta um exemplo de uso deste comando, listando o conteúdo do diretório corrente.

Listagem 20. Exemplo de utilização da estrutura condicional *case*.

```
01 #!/bin/bash
02 echo "Calculadora no shell"
03 if [ $# -ne 3 ]
04 then
05   echo "Uso: ./calculadora.sh num1 num2 operacao"
06   echo "As operações possíveis são: soma subtrai multiplica e divide"
07 else
08   case "$3" in
09     soma)
10       echo "Soma: $1 + $2 = $(expr $1 + $2)"
11       ;;
12     subtrai)
13       echo "Subtração: $1 - $2 = $(expr $1 - $2)"
14       ;;
15     multiplica)
16       echo "Multiplicação: $1 * $2 = $(expr $1 * $2)"
17       ;;
18     divide)
19       echo "Divisão: $1 / $2 = $(expr $1 \ $2)"
20       ;;
21     *)
22       echo "Digite dois números inteiros e escolha uma das operações abaixo:"
23       echo "soma subtrai multiplica e divide"
24       ;;
25   esac
26 fi
```

Listagem 21. Sintaxe do comando *for*.

```
for <var> in <list>
do
Bloco de comandos
done
```

Listagem 22. Exemplo de uso do comando *for*.

```
$ vim lista.sh

01 #!/bin/bash
02 ARQ=$(ls)
03 echo "Lista de arquivos do diretório $(pwd):"
04 for nomes in $ARQ;
05 do
06   echo $nomes
07 done
```

No exemplo, o resultado do comando *ls* é armazenado na variável chamada *ARQ*, e o comando *echo* exibe a mensagem "Lista de arquivos do diretório \$(pwd)", sendo que a variável *\$(pwd)* será convertida para o path absoluto (caminho completo) do diretório onde o usuário se encontra no momento da execução do programa, conforme é mostrado no trecho a seguir:

```
2 ARQ=$(ls)
3 echo "Lista de arquivos do diretório $(pwd):"
```

Em seguida o comando *for* é executado, e para cada arquivo que o laço lê, o nome é armazenado na variável *nomes*. Logo após é executado o bloco *do*, onde o nome do arquivo será mostrado na tela por meio do comando *echo* (linha 6). Esse processo será repetido até que o último nome de arquivo encontrado seja exibido. A construção do *for* é finalizada com o comando *done*, exibido na linha 7.

while

O laço *while* é uma estrutura de controle de fluxo que permite que um conjunto de instruções seja executado enquanto uma condição estiver satisfeita, ou seja, os comandos serão executados continuamente enquanto uma condição retornar um código 0 (verdadeira). A **Listagem 23** apresenta a sintaxe do comando *while*.

Listagem 23. Sintaxe do comando *while*.

```
while<comando>
do
Bloco de comandos
done
```

Na sintaxe apresentada, o comando é executado e seu código de retorno é testado. Caso seja igual a 0, só então o bloco de comandos é executado, e ao encontrar o comando *done*, o ciclo é reiniciado até que o comando devolva um código de retorno diferente de 0, quando então a execução do programa é finalizada. A **Listagem 24** demonstra um exemplo do comando *while*.

Listagem 24. Exemplo de uso do comando *while*.

```
#!/bin/bash
echo "Digite o nome do mascote do linux:"
read MASCOTE
while [ $MASCOTE != "tux" ]
do
echo "Nome digitado esta errado"
echo "Digite novamente o nome do mascote do linux:"
read MASCOTE
done
echo "O nome esta correto!"
```

Neste exemplo, o script lê o conteúdo da variável *MASCOTE* até que a resposta seja *tux*, ou seja, a pergunta será repetida enquanto a resposta for diferente de *tux* (*\$MASCOTE* = "*tux*"). Quando a resposta for correta, a repetição é encerrada e o script será finalizado pelo comando *done*.

Conclusão

Neste artigo apresentamos os principais conceitos sobre shell script, um item essencial para usuários e administradores de sis-

temas. O shell script permite explorar e personalizar o ambiente e combinar comandos de forma a realizar tarefas que vão da criação de arquivos até a realização de backups.

Esta rica variação demonstra a importância de se dominar o foco desse estudo. Com ele é possível automatizar tarefas e tornar nossas atividades de administração de sistemas muito mais simples.

Autora



Ivani Nascimento

ivani.araujo@gmail.com

Atua no ramo de tecnologia da informação e infraestrutura desde 2006, escreveu artigos para a revista EasyLinux e ministrou mais de 200 horas/aula de formação Linux. Graduada em Tecnologia de Redes de Computadores e Especialista em Segurança da Informação pelo ITA, atua como Analista de Suporte ao Desenvolvimento em uma empresa do mercado financeiro.



Autora



Pathiene Gerstenberger

pathigerst@gmail.com

Atua no ramo de tecnologia da informação e infraestrutura desde 2006, escreveu artigos para a EasyLinux e ministrou mais de 200 horas/aula de formação Linux. Bacharel em Ciências da Computação, MBA em Empreendedorismo e é certificada LP11-C e NCLA, atua como Analista Linux em uma empresa de catálogo de ofertas.



Links e Referências:

Robbins, A., Beebe, N. H. F. **Classic Shell Scripting**. O'Reilly. 2005.

Albing, C., Vossen, JP., Newham, C. **Bash Cookbook - Solutions and Examples for bash Users**. O'Reilly. 2007.

Lakshman, S. **Linux Shell Scripting Cookbook**. Packt Publishing. 2011.

Bash Reference Manual

http://www.gnu.org/software/bash/manual/bashref.html#What-is-Bash_003f

COOPER, M. Advanced Bash-Scripting Guide

<http://tldp.org/LDP/abs/html/index.html>

Site Oficial KornShell

<http://www.kornshell.com/>

Canivete Suíço do Shell

<http://aurelio.net/shell/canivete/>

Evolução de shells no Linux

<http://www.ibm.com/developerworks/br/library/l-linux-shells/>

Shell builtin

http://en.wikipedia.org/wiki/Shell_builtin

Transmissão de dados via rede elétrica

Tecnologia “sem novos fios” para acesso à rede banda larga

O desenvolvimento de novas tecnologias que conseguem atingir altas taxas de transmissão de dados e melhoram o desempenho das redes propiciou o aparecimento de diversos padrões de redes com fio, sem fio e sem novos fios, assim como padrões com redes mistas e padrões proprietários. Nos últimos anos, um grande esforço tem sido realizado para permitir a viabilidade da rede elétrica para a transmissão de dados em banda larga. Este esforço inclui o desenvolvimento de equipamentos para a rede de acesso, tanto em baixa quanto em média tensão, além de equipamentos para o usuário final, baseado no conceito de “aproveitamento da rede elétrica” (utilizar ao máximo o potencial do que já foi investido de infraestrutura para alcançar uma maior abrangência de atendimento).

Esta tecnologia é denominada PLC/BPL (*Power Line Communication/Broadband over Power Line Communications*), que tem como objetivo utilizar a rede elétrica para a transmissão de dados em banda larga. De maneira geral, a **Figura 1** apresenta uma topologia típica da rede PLC, com três níveis de rede: a rede de acesso, a rede de distribuição e a rede dos provedores de serviço (ponto de interconexão com a Internet).

Analizando-se apenas os canais PLC em ambientes prediais, estimativas recentes apontam que esta tecnologia suporta taxas de transmissão de dados de até 600Mbps, e comercialmente, existem equipamentos que transmitem em 200Mbps. Para um melhor nível de compatibilidade, o grupo de trabalho do IEEE denominado P.1901 definiu uma norma para controle de acesso ao meio de transmissão e especificações da camada física em redes PLC, o que facilitaria o desenvolvimento de novos equipamentos compatíveis com a tecnologia.

No Brasil, os serviços de telecomunicações tiveram um forte incentivo com o processo de privatização e regulamentação, ocorrido a partir da Emenda Constitucional nº 8 de 15 de agosto de 1995. Esta emenda abriu a possibili-

Resumo DevMan

Transmissão de dados via rede elétrica:

Este artigo descreve conceitos fundamentais sobre as tecnologias PLC (Power Line Communications), HomePlug e Smart Grid, e é motivado por um crescente interesse na aplicabilidade das linhas de energia como meio alternativo de propagação de sinais de comunicação. Com o desenvolvimento da tecnologia PLC, que permite transmissões de sinais elétricos e de dados em redes de distribuição de energia, surge mais uma opção de conectividade em banda larga, além dos sistemas existentes, como wireless, de satélite e cabos coaxiais das operadoras de TV por assinatura.

Em que situação o tema é útil:

A tecnologia PLC vem sendo desenvolvida há algum tempo e pode vir a ser uma alternativa para serviços de banda larga, devido à preexistência do cabeamento da rede de energia elétrica, além da mobilidade encontrada na implementação de redes locais, uma vez que cada tomada elétrica pode ser também um ponto de rede, aproveitando assim o cabeamento elétrico já existente para o tráfego de dados.

dade de exploração dos serviços telefônicos, telegráficos, de transmissão de dados e demais serviços públicos de telecomunicações por meio de concessão, permissão ou autorização concedida pela União a empresas privadas, nos termos da lei, conhecida como Lei Geral de Telecomunicações (LGT) ou Lei nº 9.472/97.

Com o setor de telecomunicações passando por um crescimento exponencial, muitas empresas elétricas começaram a pensar em se tornar provedores de serviços de telecomunicações na década de 90.

Para a Agência Nacional de Energia Elétrica (ANEEL), “Power Line Communication (PLC) é um tipo de sistema que permite a transmissão de sinais de internet, voz, vídeo e comunicação digital e analógica, por meio da rede elétrica”. O mercado de distribuição de energia elétrica é atendido por 64 concessionárias, estatais ou privadas, que abrangem todo o País, que atendem cerca de 60 milhões de unidades consumidoras, das quais 85% são consu-

midores residenciais, em mais de 98% dos municípios brasileiros. O Estado de Goiás, por exemplo, possui mais de dois milhões de unidades consumidoras de energia elétrica, distribuídos em 237 municípios e beneficiando, aproximadamente, quatro milhões de habitantes (cerca de 90 % da população total do Estado). As adversidades do meio elétrico são as principais barreiras para a transmissão de dados nesse tipo de canal.

Pode-se afirmar que a tecnologia PLC possibilita desenvolver diferentes aplicações, gerando, como consequência, inovações tecnológicas. A utilização da rede PLC tem como principais vantagens:

- Infraestrutura de cabeamento pronta;
- Alta taxa de transmissão de dados;
- Desenvolvimento de novas técnicas de transmissão digital de dados;
- Solução last meters (transformação da rede elétrica de distribuição predial ou residencial em uma rede local de dados);
- Facilidade de uso e na implementação do sistema PLC. Cada tomada é um ponto de acesso de uso simples e descomplicado;
- Alternativa de acesso à banda larga promovendo a competição no segmento de prestação de serviço e inclusão digital.

Através de mecanismos de gerência, planejamento e operação dos sistemas de geração, distribuição e transmissão de energia elétrica, utilizando uma abordagem coordenada e colaborativa, é possível que a tecnologia PLC venha a se tornar um meio de transmissão viável para acesso a banda larga. Além disso, a tecnologia PLC/BPL permite que uma série de serviços seja oferecida por concessionárias de energia elétrica, entidades públicas e empresas de telecomunicações, como mostra a **Figura 2**.

No Brasil, as concessionárias de energia elétrica realizaram nos últimos anos alguns testes para também oferecer serviços de telecomunicações. Atualmente, o foco dessas concessionárias é o estudo do Smart Grid, que propõe a convergência entre o sistema de energia elétrica e o de comunicação de dados. O Smart Grid é um sistema de T&D (Transmissão e Distribuição) que possibilita a disponibilidade da informação no local e

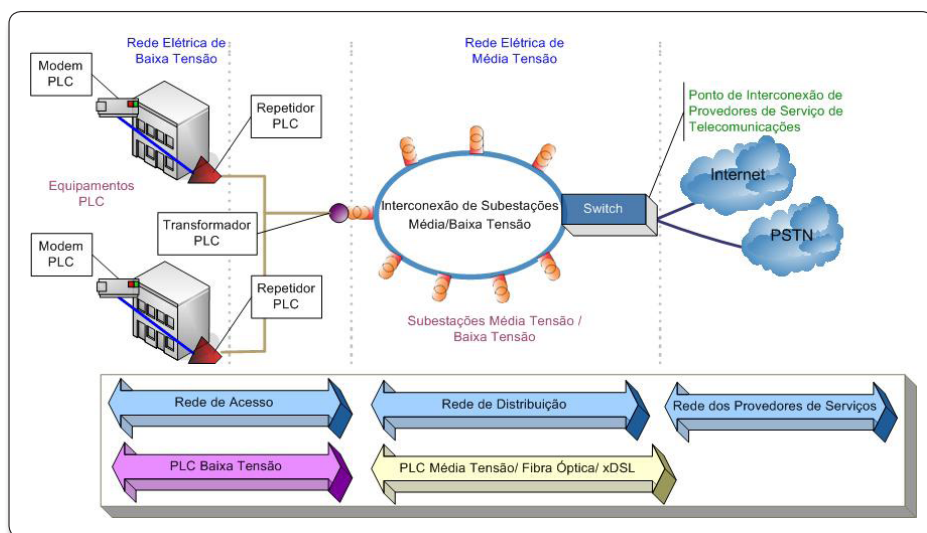


Figura 1. Topologia típica da rede PLC: rede de acesso, rede de distribuição e rede dos provedores de serviços de telecomunicações

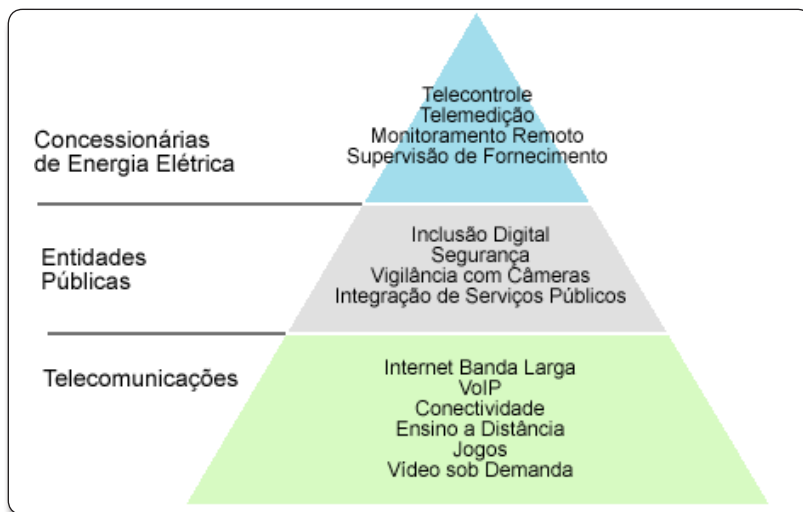


Figura 2. Possíveis aplicações para a tecnologia PLC

momento desejado, tornando-o assim uma tecnologia de rede elétrica inteligente que é definida principalmente pelas funcionalidades que possui.

Características da tecnologia PLC

A tecnologia PLC não pode ser considerada uma nova tecnologia, pois desde o início do século XX as redes elétricas têm sido utilizadas internamente pelas empresas de energia elétrica para suportar serviços de telecomunicações. A princípio, no entanto, ela não havia sido considerada uma forma de comunicação viável devido a sua baixa taxa de transmissão, baixa funcionalidade e alto custo de desenvolvimento.

Apesar disso, desde a década de 1920 os Sistemas Power Line Carrier vêm sendo utilizados pelas empresas de energia elétrica, em telemetria, controle remoto e comunicação de voz. A primeira técnica que possibilitou a utilização da rede de distribuição de energia elétrica para a transmissão de alguns sinais de controle e monitoramento da rede elétrica foi desenvolvida no início da década de 1950, conhecida como Ripple Control. Esse método era caracterizado pela utilização de baixas frequências (entre 100Hz e 900Hz), possibilitando comunicação a taxas bem baixas e necessitando de alta potência para transmissão. A comunicação era unidirecional, sendo utilizada

para tarefas simples como acionamento da iluminação pública, tarifação e controle do limite de carga suportado pelas linhas de transmissão, para evitar a queima por sobrecarga dos equipamentos elétricos dos usuários.

Apenas na década de 1990 foram obtidos sistemas capazes de fornecer comunicação de forma bidirecional através da rede de distribuição, sendo marcados pela utilização de frequências mais elevadas e menores níveis de potência transmitida. Em 1991, Dr. Paul Brown, da Norweb Communications, iniciou alguns testes com comunicação digital de alta velocidade utilizando linhas de energia. Entre 1995 e 1997 foi demonstrada a possibilidade de se resolver problemas de ruído e interferência, permitindo a viabilidade da transmissão de dados de alta velocidade. Em outubro de 1997, a Nortel e Norweb fizeram o primeiro teste de acesso à Internet em uma escola de Manchester, e na Europa foi criado o PLC Fórum. Em 1998 a UTC (Utilities Telecommunications Council) lançou nos Estados Unidos o PowerLine Telecommunications Forum (PLTF).

Em se tratando da evolução das taxas de transmissão de acordo com a frequência utilizada, tem-se:

- Década de 80: até 144 kbps (< 500kHz) – OPLAT/CARRIER;
- Início da década de 90: até 10 Mbps (<10Mhz);
- Início do século XXI: até 45 Mbps (<30Mhz);
- Atualmente: até 204 Mbps (1-30 Mhz);
- Futuro: >2 Gbps (1-500 Mhz).

Mesmo com todo esse desenvolvimento, o meio físico do PLC ainda é bastante hostil para os dados, visto que não foi concebido para este fim. Quando se fala de redes elétricas, elas são classificadas em três níveis: (100kV) Alta Tensão, (1-100kV) Média Tensão e (1kV) Baixa Tensão, cada qual adaptado para interligar diferentes distâncias. O sinal PLC é transmitido sobre os fios de cobre (ou alumínio) das redes de distribuição de baixa e média tensão, e algumas propriedades das redes de energia elétrica influenciam negativamente nas comunicações em alta velocidade (perdas no cabo, propagação em múltiplos caminhos e ruído são alguns exemplos). Uma forma de reduzir o impacto das características do meio de transmissão na comunicação é a aplicação de métodos eficientes de modulação, como o OFDM (*Orthogonal Frequency Division Multiplexing*), além de mecanismos de correção de erro como o FEC (*Forward Error Correction*) e o ARQ (*Automatic Repeat reQuest*).

Em redes PLC é necessário utilizar alguns dispositivos específicos para que a rede elétrica possa ser usada para trafegar dados, como pode ser visto na **Figura 3**.

No caso das redes PLC, para se introduzir e adaptar os sinais de dados dos equipamentos para as redes de Baixa e Média Tensão que chegam até as residências dos usuários são necessários acopladores, que podem ser capacitivos (injetam os sinais de dados através de contato direto com as linhas de energia elétrica) ou indutivos (injetam os sinais na rede elétrica por indução). Os acopladores do tipo indutivo são mais utilizados em média

tensão, enquanto os acopladores capacitivos têm maior aplicação na injeção do sinal PLC em baixa tensão, e o Master PLC, conhecido como “head end” da rede, é instalado próximo ao transformador de baixa tensão e tem a função de gerenciar e rotear informações aos repetidores, conhecidos como slave nodes.



Figura 3. Dispositivos da Rede PLC

Esses repetidores regeneram e reinjetam o sinal PLC proveniente dos Equipamentos de Transformador (placas instaladas em transformadores de média tensão/baixa tensão da rede elétrica, que extraem o sinal do canal que distribui a tecnologia PLC e o injeta na rede de baixa tensão) para a rede elétrica de distribuição doméstica. Os repetidores são instalados normalmente junto aos medidores de cada prédio ou em algum local intermediário, como por exemplo, postes sem transformador, na rede de distribuição de baixa tensão. Em alguns casos, não é necessária a utilização de repetidores (por exemplo, onde o equipamento PLC do Transformador consegue uma conexão de elevada qualidade com o Modem PLC).

O Modem PLC, por sua vez, é conectado à tomada de energia da rede de distribuição, por onde, além de ser alimentado, recebe o sinal PLC através de uma tomada simples. Neste modem existe um filtro, parecido com o que se utiliza em redes xDSL, mas que filtra sinais de dados e sinais elétricos. Para permitir a integração das redes PLC com serviços de telecomunicações, esse modem PLC normalmente possui interfaces de comunicação RJ45 para a rede Ethernet, USB (*Universal Serial Bus*) e interface RJ11, que permite a conexão de aparelhos telefônicos.

De maneira geral, na estrutura da rede PLC, de acordo com especificações dos principais fabricantes, cada repetidor gerenciado por um Master possui a capacidade de gerenciamento de 63 equipamentos “escravos”, formando uma topologia de comunicação com aproximadamente 4.000 usuários por transformador, e a distância média entre repetidores de 1.500 metros, aumentando

o custo de implementação em distâncias maiores. Em relação à segurança, uma rede PLC é considerada isolada de outra quando não existe uma conexão física entre elas e quando o sinal de uma rede não é percebido pela outra, seja devido à atenuação do sinal ou ao uso de fases elétricas distintas.

Para garantir o serviço de voz e dados a clientes extremamente remotos, empresas como a Copel Telecomunicações muitas vezes utilizam a tecnologia PLC. Esses clientes normalmente têm acesso ao serviço de energia elétrica, mas não possuem acesso ao serviço de telefonia convencional devido ao alto custo para fornecer esse tipo de serviço com cabeamento de cobre. A **Figura 4** apresenta o esquema de uma cidade utilizando a rede elétrica para transmissão de dados, com linhas prioritárias para serviços de utilidade pública, como bombeiros, hospitais, postos policiais, entre outros. Essas linhas prioritárias garantem que esses serviços não seriam prejudicados pelo cabeamento da tecnologia PLC.

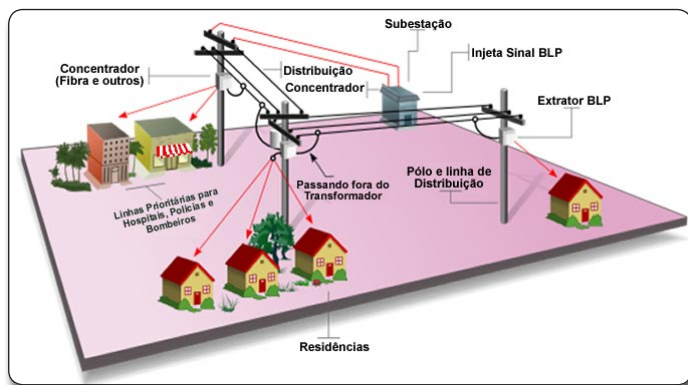


Figura 4. Esquema de uma cidade utilizando PLC/BLP

Ambientes Indoor – HomePlug

Em ambientes internos (indoor), residenciais ou empresariais, uma alternativa interessante é o padrão HomePlug, que dispensa a necessidade de se colocar na rede elétrica o equipamento denominado Master. O padrão HomePlug foi desenvolvido pela HomePlug Powerline Alliance, definindo seu método de acesso ao meio de transmissão e especificações da camada física para redes de baixa tensão (ambientes residenciais, por exemplo).

Atualmente, o principal concorrente do HomePlug é o padrão IEEE 802.11 (redes sem fio), devido ao seu sucesso comercial, mas existem diversos testes de desempenho comparativos entre os dois padrões em cenários domiciliares para mostrar a viabilidade do HomePlug como alternativa de tecnologia de ponto acesso (Intranet). Contudo, a distribuição da Internet está sujeita a contratação do serviço através de alguma empresa de telecomunicações.

No padrão HomePlug, dois ou mais computadores são ligados em rede simplesmente conectando-se a eles um adaptador, como o da **Figura 5**, que por sua vez é conectado a uma tomada elétrica. Para possibilitar essa conexão, faz-se necessário que as tomadas onde os adaptadores são ligados pertençam à mesma fase do quadro de distribuição de energia.



Figura 5. Adaptador HomePlug

O HomePlug foi introduzido no mercado consumidor dos Estados Unidos em maio de 2002 e, aproveitando-se da grande disponibilidade de tomadas existentes na maioria das edificações, tornou-se bastante difundido em casas e pequenos escritórios. Este padrão possibilita que os computadores ligados através dele compartilhem os recursos de rede, como impressoras, ou até mesmo a conexão com a Internet, que pode ser feita através de outro tipo de tecnologia, como ADSL ou Cabo. As taxas de transmissão no padrão HomePlug podem chegar a 200Mbps (HomePlug AV), sendo comparáveis com as taxas de transmissão obtidas no padrão de redes wireless IEEE 802.11.

Dependendo do fabricante, os equipamentos HomePlug podem ter alcance de cerca de 150 metros a 300 metros. A meta do HomePlug AV é permitir a distribuição de transmissões de áudio e vídeo em alta definição (HDTV). Esse novo padrão foi desenvolvido pela Panasonic, Intellon, DS2, Sharp e Conexant, e possibilita uma taxa útil de dados de 150Mbps para comunicações mais robustas em canais ruidosos, como o meio elétrico.

Sistema Elétrico Inteligente – Smart Grid

Smart Grid é o sistema elétrico inteligente que monitora em tempo real o comportamento de usuários e elementos da rede, permitindo um melhor aproveitamento dos recursos da rede e fornecer eficientemente uma energia sustentável, econômica e segura.

O Smart Grid promete diversas melhorias para o uso da energia elétrica. Com um medidor mais informativo e redes de distribuição inteligentes, será possível economizar na conta de luz e ainda verificar os dispositivos que mais consomem energia em residências e estabelecimentos comerciais. Diferentemente do PLC de banda larga, o Smart Grid com PLC de banda estreita utiliza frequências diferentes e teoricamente não causa problemas para o Serviço de Radioamador e demais usuários dos serviços licenciados de rádio das médias e altas frequências, como serviços de comunicações marítimos e aeronáuticos. Um dos grandes problemas para a utilização do PLC é justamente a interferência que a tecnologia causa em outros serviços que utilizam a mesma frequência ou frequências próximas.

A importância de se usar Smart Grid encontra-se na adição de vários serviços, que podem ser úteis tanto aos consumidores quanto às concessionárias, entidades públicas e operadoras de telecomunicações. Serviços esses como: controle, medição e supervisão à distância de energia elétrica, adicionado a serviços de corta/religa remoto (como acontece com os serviços de telecomunicações ao ativar/desativar um determinado serviço na linha do consumidor no caso de ausência de pagamento da conta); integração de serviços básicos como energia, telecomunicações, água e gás em uma única conta; segurança pública (câmeras IP) e inclusão digital através do acesso a Internet.

Para se ter na prática uma rede elétrica inteligente são necessárias algumas modificações, inclusive físicas, como é o caso da substituição dos medidores eletromecânicos por medidores mais avançados. Medidores esses que são capazes de uma comunicação nos dois sentidos, tanto recebendo como enviando dados, fornecendo assim serviços mais avançados e completos aos clientes e também em benefício da operadora; serviços que irão além da simples função de leitura. Serão necessários também outros equipamentos de energia mais avançados, como é o caso de um Concentrador PLC, que deverá ficar entre a linha de baixa tensão (110/220 V) e o medidor inteligente na propriedade do usuário, fazendo assim uma comunicação entre o medidor inteligente e o transformador, sendo chamado também de Medição Externa Centralizada (MEC). O MEC é considerado um Gateway Universal na rede, tendo a função de Centro de Controle, sendo responsável por enviar as informações através do transformador à linha de distribuição, além dos serviços remotos como medição da energia ativa e serviço corta/religa.

Os medidores presentes nas propriedades dos usuários terão como função principal a medição da energia consumida. Outros serviços como controle de qualidade e serviço de corta/religa serão executados pelo MEC, com a intenção de evitar fraudes e possíveis artimanhas com relação à energia elétrica.

Os transformadores também terão um importante papel nessa rede inteligente, com a função de monitoramento da energia transmitida, monitorando a tensão, verificando sobrecargas e controle de perdas, enviando os dados automaticamente pela rede de distribuição à subestação para possíveis reparos.

As redes de distribuição e subestações serão automatizadas para receber informações e possibilitar os reparos em tempo hábil, aumentando a eficiência e proporcionando uma auto reconfiguração com a utilização de chaves automatizadas.

O Smart Grid já é realidade em algumas partes do mundo, como, por exemplo, nos Estados Unidos, país com o maior número de concessionárias envolvidas em pesquisas e testes. No Brasil, existem várias pesquisas em desenvolvimento, e recentemente, o governo federal lançou uma linha de financiamento para projetos de inovação tecnológica no setor elétrico. Chamado de Inova Energia, o programa terá ênfase em três áreas: redes inteligentes de energia (Smart Grids), fontes alternativas de energia e veículos elétricos.

As concessionárias brasileiras envolvidas em sistemas de Smart Grid buscam as mesmas finalidades como telecontrole, teleme-

dição, monitoração remota, liga/religa remoto e supervisão de fornecimento, com o intuito de prover serviços mais avançados aos consumidores, além de melhorias internas como controle da energia distribuída e redução dos custos de operação.

Os benefícios de uma rede inteligente atingirão as concessionárias, entidades públicas e empresas de telecomunicações. Lógico que todos os benefícios terão como alvo final os usuários, mas intermediados por organizações diferentes. As entidades públicas serão beneficiadas com avanços tecnológicos nos serviços prestados, como é o caso do programa de inclusão digital (banda larga com o uso em potencial da tecnologia PLC) e também melhorias em serviços como segurança, vigilância através de câmeras e integração dos serviços públicos, sendo que todos esses benefícios tornarão todo o sistema mais rápido e seguro. As operadoras de telecomunicações poderão prover acesso à internet banda larga em uma infraestrutura previamente existente, além de melhorias em serviços já realizados como VoIP e vídeo em demanda, que necessitam de um sistema mais veloz e seguro.

Com a evolução do Smart Grid e sua implantação, várias outras utilidades e serviços extras estarão direta e indiretamente sendo beneficiados com a utilização do PLC. Uma vantagem do uso de PLC é a criação de uma rede interna denominada HAN (*Home Area Network*), com funções adicionais como medição interna de consumo de cada aparelho eletroeletrônico, monitoramento da qualidade da energia recebida, além da integração com outras tecnologias já existentes como Bluetooth, HomePlug, WiFi, ZigBee, entre outros. Com essa utilização em Smart Grid, a HAN se tornaria um sistema híbrido, com duas ou mais tecnologias para comunicação de dados, o que já está sendo muito difundido atualmente.

A ideia de transmitir dados sobre a rede elétrica também pode ser aplicada para interconectar dispositivos inteligentes dentro de uma casa. No início de 2000, a empresa Sunbeam – por meio de sua subsidiária Thalia Products – anunciou uma linha de eletrodomésticos inteligentes que trocavam informações no momento em que eram ligados à tomada. Batizada de HLT (*Home Linking Technology*), a iniciativa pretendia lançar produtos como despertadores, detectores de fumaça, cafeteiras, cobertores elétricos, medidores de pressão arterial, entre outros, capazes de se comunicar. Por exemplo, o despertador poderia ser programado para mandar uma ordem à cafeteira para começar a preparar o café um pouco antes de uma pessoa levantar da cama.

Segurança e Gerência em Sistemas PLC

Uma rede PLC pode variar em tamanho, complexidade e ser constituída de diferentes dispositivos, inclusive ligando redes distintas. Alguns equipamentos PLC são capazes de utilizar o protocolo SNMP (*Simple Network Management Protocol*) para ser possível um gerenciamento centralizado, utilizando MIBs (*Management Information Base*) proprietárias que permitem maior controle sobre os equipamentos ativos da rede PLC.

Como em uma rede PLC o meio de transmissão é compartilhado entre todas as residências atendidas pelo transformador, pode ser possível escutar transmissões alheias. Para evitar esse tipo

de problema, a tecnologia PLC permite diversos mecanismos de segurança, como criptografia, autenticação e redes locais virtuais (VLANs). Esses equipamentos PLC criam redes separadas em um mesmo circuito elétrico com duas chaves de cifragem diferentes, configuráveis através de um programa fornecido geralmente com o equipamento.

A criptografia na rede PLC acontece na camada de enlace, garantindo confidencialidade ao enlace estabelecido pelo usuário. A autenticação ocorre no momento em que um equipamento tenta se associar à rede e auxilia o controle de acesso à rede e a utilização de seus recursos.

Grande parte dos dispositivos PLC opera com o algoritmo de criptografia DES (*Data Encryption Standard*) de 56 bits, e possui sistema de detecção de intrusos, para que nenhum acesso seja feito sem o conhecimento da administração da rede. Equipamentos mais novos, que operam no padrão HomePlug AV, utilizam criptografia AES (*Advanced Encryption Standard*) de 128 bits, que é baseado no DES (**BOX 1**), e suportam mudanças de chaves de criptografia dinâmicas. Essas mudanças permitem que uma estação participe de múltiplas redes HomePlug AV.

Para garantir a confidencialidade, integridade, disponibilidade e controle de acessos não autorizados, os seguintes itens de segurança devem ser verificados:

- a) Vulnerabilidades na rede elétrica;
- b) Controle de criptografia;
- c) Proteção contra softwares maliciosos;
- d) Controle de acesso à rede;
- e) Controle de acesso ao sistema operacional.

BOX 1. Algoritmo DES

O Algoritmo DES foi criado pela IBM e implantado pelos EUA em 1977 para criptografia padrão de informações não-confidenciais, e é considerado por muitos autores como sendo uma criptografia não muito segura, apesar de rápida. Foi quebrado por "força bruta" em 1997 em um desafio lançado na Internet.

Conclusão

O surgimento da eletricidade como uma tecnologia foi uma das maiores descobertas do homem. Com ela, a humanidade produziu mais, viveu melhor e inventou mais coisas. Mas seu uso ficou praticamente estagnado no tempo após sua evolução inicial.

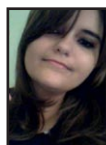
A necessidade da sociedade por serviços de telecomunicações, principalmente por serviços de transmissão de dados em banda larga, é o fator motivador para que grandes esforços sejam realizados no campo de pesquisa para viabilizar a utilização das tecnologias PLC em banda larga e Smart Grid.

A adoção em larga escala da tecnologia PLC tem ainda como desafios a falta de manutenção de algumas redes de energia

elétrica e a utilização clandestina da energia elétrica pública, com perigosas conexões piratas que afetarão o funcionamento da tecnologia PLC no Brasil.

De acordo com o apresentado, as tecnologias PLC, HomePlug e Smart Grid são complementares, tendem a evoluir juntas e serem utilizadas em conjunto em um futuro próximo. Por meio de mecanismos de gerência, planejamento e operação dos sistemas de geração, distribuição e transmissão de energia elétrica, é possível que a tecnologia PLC venha a se tornar uma solução das chamadas tecnologias de última milha, permitindo o surgimento de uma série de serviços que poderá ser oferecida por concessionárias de energia elétrica, entidades públicas e provedores de serviços de telecomunicações.

Autor



Christiane Borges Santos

christiane.borges@ifg.edu.br

Atua como docente na área de Redes de Computadores há mais de cinco anos. É tecnóloga em Redes de Comunicação, pelo CEFET Goiás, mestre em Engenharia Elétrica e da Computação, pela UFG. Atualmente cursa doutorado em Ciências da Computação, na UnB. É professora adjunta no IFG e membro do comitê gestor do grupo /MNT (Mulheres na Tecnologia).



Links:

ANATEL – Agência Nacional de Telecomunicações. Resolução Nº 527

<http://www.anatel.gov.br>

ANEEL – Agência Nacional de Energia Elétrica. Resolução Normativa Nº 375

<http://www.aneel.gov.br>

Copel Telecomunicações

<http://www.copel.com/>

HomePlug Powerline Alliance - White Papers

<http://www.HomePlug.org>

IEEE Communications Society Technical Committee on Power Line Communications

<http://committees.comsoc.org/plc/>

Low-Frequency Narrow-Band Power Line Communications Working Group

<http://grouper.ieee.org/groups/1901/2/>

Smart Grid Light - Redes Inteligentes

<http://SmartGridLight.com.br/conceitos-smart-grid>

The ABB Group – Soluções em Redes PLC Média Tensão.

<http://www.abb.com>

Tutoriais Banda Larga e PLC

<http://www.teleco.com.br/tutoriais/tutorialplc>

Migração de dados: Metodologias e abordagens host based – Parte 2

Conheça os procedimentos operacionais utilizados para migração de dados corporativos em ambiente Linux



ESTE ARTIGO FAZ PARTE DE UM CURSO

Conforme visto na primeira parte deste artigo na edição anterior da revista Infra Magazine, o processo de migração de dados corporativos contempla diversos procedimentos operacionais e várias tarefas administrativas relacionadas ao planejamento da atividade, tornando a etapa de “Movimentação dos Dados” em si apenas parte ou uma pequena fração do processo.

Diversos outros fatores como planejamento, definição de qual estratégia utilizar levando em consideração risco x segurança x conforto operacional, direcionamentos do negócio, janela produtiva, procedimentos de Rollback, comunicação interna, entre outros, devem ser respeitados e estar alinhados com as necessidades do negócio para se decidir com objetividade, de fato, “Qual a melhor estratégia de movimentação” a ser utilizada para migrar determinado dado ou informação.

Vimos também um dos principais pilares para decisão de qual estratégia de migração utilizar, que é a situação ou forma em que o dado se encontra hospedado. O dado encontra-se armazenado em um raw device? O *File System* é dinâmico? A aplicação tem alguma peculiaridade? Existe algum gerenciador de volumes? Existe Multipath? Essas, entre outras questões, interferem no processo de

Resumo DevMan

Migração de dados: Metodologias e abordagens host based – Parte 2:

Neste artigo serão apresentadas algumas estratégias e metodologias utilizadas para migração de dados corporativos de ambientes Enterprise Linux adotadas pelo mercado corporativo de TI, mais especificamente o segmento Datacenter, abordando os procedimentos operacionais para efetuar a migração empregando as técnicas definidas na primeira parte deste artigo. Utilizaremos recursos do próprio gerenciador de volumes LVM e ferramentas nativas do sistema operacional Linux como RSYNC e TAR.

Em que situação o tema é útil:

O artigo tem como público alvo a geração Datacenter, como técnicos e administradores de TI responsáveis por executar ou planejar migrações de dados corporativos, envolvendo ambientes de missão crítica compostos por sistemas operacionais Linux.

definição do método de movimentação e também podem viabilizar ou não determinada estratégia de migração.

Na primeira parte deste artigo, foram apresentados como exemplos três servidores, sendo estes RJSRVUXX01, RJSRVUXX02 e RJSRVUXX03, respectivamente. Cada um destes servidores possui seus *File Systems* hospedados em situações específicas de forma que, para realizar a migração de dados de cada um desses

File Systems, foi escolhida uma técnica distinta de movimentação de dados, variando entre estratégias utilizando recursos do próprio gerenciador de volumes LVM ou ferramentas nativas do sistema operacional, como RSYNC e TAR.

Agora, na segunda parte do artigo, abordaremos em detalhes como executar essas estratégias, apresentando todos os comandos e procedimentos operacionais para execução de cada uma das migrações de dados.

Procedimento Operacional

A partir de agora, serão apresentadas as três estratégias de migração abordadas no artigo. Antes de efetuar a migração dos dados em si, sempre é recomendado verificar se existem backups confiáveis e íntegros de todos os dados envolvidos na migração.

Conforme definido na primeira parte do artigo, cada um dos três servidores do nosso ambiente terá uma estratégia de migração específica em função de como os dados encontram-se hospedados. As estratégias que serão empregadas para migração dos dados são:

1. Servidor RJOSSRVUXX01 – Estratégia LVM PV Move;
2. Servidor RJOSSRVUXX02 – Estratégia LVM Split Mirror;
3. Servidor RJOSSRVUXX03 – Estratégia RSYNC.

Migração do servidor RJOSSRVUXX01 – Estratégia LVM PV Move

Conforme visto na **Figura 1**, responsável por consolidar o mapeamento realizado no servidor RJOSSRVUXX01, esse host tem quatro LVOLs (Logical Volumes) a serem migrados atualmente hospedados em três PVs (Physical Volumes). Nesse procedimento, criaremos três novos PVs e depois efetuaremos a operação “Movimentação de PV” para realizar a migração dos dados.

A **Figura 2** ilustra os estágios que ocorrerem durante a fase de movimentação dos dados utilizando a estratégia “LVM PV Move”. Cada um dos estágios será comentado durante o procedimento encontrado na **Listagem 1**, utilizada para acompanhar

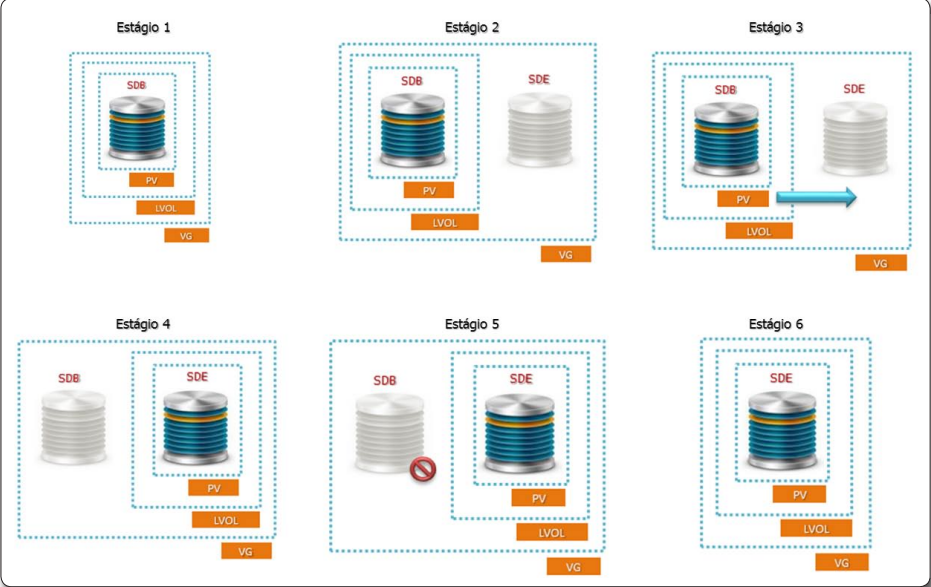


Figura 1. Dados a serem migrados do servidor RJOSSRVUXX01

FileSystem	Device	Tamanho (Gb)	FS	Volume Manager	VG	LV	PE	PE Size
/srv/db/MYDB	/dev/db_vg/lvol_01	10,00	ext3	LVM	db_vg	lvol_01	2560	4 M
/srv/db/data01	/dev/db_vg/lvol_02	50,00	ext3	LVM	db_vg	lvol_02	12800	4 M
/srv/db/indx01	/dev/db_vg/lvol_03	50,00	ext3	LVM	db_vg	lvol_03	12800	4 M
/srv/db/arch	/dev/db_vg/lvol_04	10,00	ext3	LVM	db_vg	lvol_04	2555	4 M

Figura 2. Estágios do procedimento “LVM – PV Move” executado no servidor RJOSSRVUXX01

HOST	VG	Old Dev	New Dev	PE Size	PE	SIZE
SRVUXX01	db_vg	/dev/sdb1	/dev/sde1	4 M	10239	40 GB
		/dev/sdc1	/dev/sdf1	4 M	10239	40 GB
		/dev/sdd1	/dev/sdg1	4 M	10239	40 GB

Tabela 1. Tabela De/Para preenchida do servidor RJOSSRVUXX01

mento do processo de migração de dados do servidor RJOSSRVUXX01.

Para facilitar a compreensão da sequência de comandos exibida na **Listagem 1**, a dividimos em várias partes e agora explicaremos cada uma separadamente:

1. Primeiramente, adicione no servidor os novos discos que receberão os dados oriundos do processo de migração. Em nosso exemplo, os novos discos já foram reconhecidos no sistema operacional com os caminhos `/dev/sde (4:0:0:0)`, `/dev/sdf (5:0:0:0)` e `/dev/sdg (6:0:0:0)`, respectivamente.

Depois dos discos já reconhecidos no sistema operacional, preencha a coluna

“New Dev” da tabela “De/Para” inserindo os novos discos/LUNs reconhecidos, conforme destacado em negrito e observado na **Tabela 1**;

2. Entre no utilitário **FDISK** e crie uma partição de tamanho igual ou superior à `/dev/sdb1` em cada um dos novos discos. Adicionalmente, não se esqueça de colocar o tipo da partição como “8E”, senão não será possível executar os próximos passos do tutorial e não será possível criar um PV (Physical Volume) nessa partição física. Repita essa operação para os três novos discos/LUNs `/dev/sde`, `/dev/sdf` e `/dev/sdg`;

3. Visualize a estrutura LVM que será migrada conforme o Estágio 1 da **Figura 2**. Para facilitar a compreensão do procedimento que será executado, o PV existente no device `/dev/sdb1` será movimentado para o device `/dev/sde1`, o PV existente no device `/dev/sdc1` será movimentado para o device `/dev/sdf1` e o PV existente no device `/dev/sdd1` será movimentado para o device `/dev/sdg1`.

Antes que se possa movimentar o Physical Volume, primeiro devemos criar os Physical Volumes nas partições que preparamos no passo 2 (`/dev/sde1`, `/dev/sdf1` e `/dev/sdg1`);

4. Agora adicione os três novos PVs criados no passo anterior ao VG (*Volume Group*) já existente. Após conclusão desse passo, nossa estrutura LVM estará semelhante ao Estágio 2 da **Figura 2**;

5. Com a etapa preparatória da estrutura LVM concluída com sucesso, podemos dar início ao processo de movimentação dos dados

utilizando o comando `pvmove`. Essa operação pode ser executada com a aplicação em estado ONLINE. Assim que o comando for executado, a cópia já é iniciada. Repita essa etapa para cada um dos três PVs. Durante a fase de cópia, nossa estrutura LVM estará semelhante ao estágio 3 da **Figura 2**.

Assim que o comando `pvmove` concluir sua execução, o cenário esperado é que todos os Extents (áreas de 4 MB) tenham sido migrados com sucesso entre o PV origem e o PV destino e a estrutura LVM estaria semelhante ao estágio 4 da **Figura 2**;

6. Após ter concluído o processo de migração, daremos início à remoção lógica da estrutura LVM existente nos discos/LUNs antigos (`/dev/sdc`, `/dev/sdd` e `/dev/sde`), pois não precisaremos mais desses discos antigos. Todos os dados já foram migrados com sucesso para os novos discos. Essa remoção é realizada utilizando o co-

Listagem 1. Sequência de comandos para migração do host RJOSRVUXX01.

```
##### Trecho 1 #####
sysadmin@rjosrvuux01:~$ lsblk --size
[0:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sda 8.58GB
[1:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sdb 42.9GB
[2:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sdc 42.9GB
[3:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sdd 42.9GB
[4:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sde 42.9GB
[5:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sdf 42.9GB
[6:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sdg 42.9GB

##### Trecho 2 #####
sysadmin@rjosrvuux01:~$ sudo fdisk /dev/sde

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-167772159, default 2048): <enter>
Using default value 2048
Last sector, +sectors or +size[K,M,G] (2048-167772159, default 167772159): <enter>
Using default value 167772159

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 8e

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

sysadmin@rjosrvuux01:~$ sudo fdisk /dev/sdf
sysadmin@rjosrvuux01:~$ sudo fdisk /dev/sdg

##### Trecho 3 #####
sysadmin@rjosrvuux01:~$ sudo pvcreate /dev/sde1
Physical volume "/dev/sde1" successfully created

sysadmin@rjosrvuux01:~$ sudo pvcreate /dev/sdf1
Physical volume "/dev/sdf1" successfully created

sysadmin@rjosrvuux01:~$ sudo pvcreate /dev/sdg1
Physical volume "/dev/sdg1" successfully created

##### Trecho 4 #####
sysadmin@rjosrvuux01:~$ sudo vgextend app_vg /dev/sde1
Volume group "app_vg" successfully extended

sysadmin@rjosrvuux01:~$ sudo vgextend app_vg /dev/sdf1
Volume group "app_vg" successfully extended

sysadmin@rjosrvuux01:~$ sudo vgextend app_vg /dev/sdg1
Volume group "app_vg" successfully extended

##### Trecho 5 #####
sysadmin@rjosrvuux01:~$ sudo pvmove /dev/sdb1 /dev/sde1
/dev/sdb1: Moved: 0.6%
/dev/sdb1: Moved: 50.0%
/dev/sdb1: Moved: 75.0%
/dev/sdb1: Moved: 88.8%
/dev/sdb1: Moved: 100.0%

sysadmin@rjosrvuux01:~$ sudo pvmove /dev/sdc1 /dev/sdf1
sysadmin@rjosrvuux01:~$ sudo pvmove /dev/sdd1 /dev/sdg1

##### Trecho 6 #####
sysadmin@rjosrvuux01:~$ sudo vgreduce db_vg /dev/sdb1
Removed "/dev/sdb1" from volume group "db_vg"

sysadmin@rjosrvuux01:~$ sudo vgreduce db_vg /dev/sdc1
Removed "/dev/sdc1" from volume group "db_vg"

sysadmin@rjosrvuux01:~$ sudo vgreduce db_vg /dev/sdd1
Removed "/dev/sdd1" from volume group "db_vg"

##### Trecho 7 #####
sysadmin@rjosrvuux01:~$ sudo pvremove /dev/sdb1
Labels on physical volume "/dev/sdb1" successfully wiped

sysadmin@rjosrvuux01:~$ sudo pvremove /dev/sdc1
Labels on physical volume "/dev/sdc1" successfully wiped

sysadmin@rjosrvuux01:~$ sudo pvremove /dev/sdd1
Labels on physical volume "/dev/sdd1" successfully wiped

##### Trecho 8 #####
sysadmin@rjosrvuux01:~$ sudo pvs
PV      VG      Fmt Attr PSize PFree
/dev/sde1 db_vg  lvm2 a- 40.00g 8.00m
/dev/sdf1 db_vg  lvm2 a- 40.00g 0
/dev/sdg1 db_vg  lvm2 a- 40.00g 0
```

mando *vgreduce*, responsável por remover um PV de um VG. Ao final do processo, nossa estrutura LVM estará equivalente ao estágio 5 da **Figura 2**.

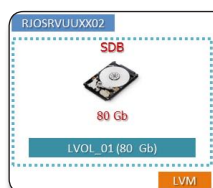
Após a remoção dos PVs antigos do Volume Group *db_vg*, nossa estrutura LVM estará semelhante ao Estágio 6 da **Figura 2**, ou seja, já sem nenhuma referência dos discos/LUNs antigos que serão removidos; 7. Para manter a consistência das informações, é importante, além de remover os PVs não mais utilizados do VG, remover os PVs das partições antigas para que o LVM nunca mais identifique ou reconheça essas áreas como PVs válidos. Utilize o comando *pvremove* para remover esses Metadados dos discos que serão removidos em um momento futuro adiante no artigo; 8. A partir desse ponto, os discos/LUNs antigos (*/dev/sdb*, */dev/sdc* e */dev/sdd*) já podem ser removidos do host/sistema operacional sem causar nenhum impacto para as aplicações. Por meio do comando *pvs*, confirme que não existe mais nenhum PV nos discos antigos além dos que devem existir de fato nos novos discos. Observe que já não existe mais nenhuma referência aos PVs que existiam nos discos SDB, SDC e SDD.

Os procedimentos de remoção de devices incluem reconfiguração da camada Multipathing no sistema operacional, remoção de Zones nos Switches SAN, Unmasking dos devices na interface de gerenciamento do Storage, Unmapping dos devices das portas do Storage na interface de gerenciamento do Storage, desfazer as LUNs no Storage, entre outros. Como esses procedimentos são muito específicos em função da tecnologia e do fabricante utilizados, não serão abordados nesse artigo.

Migração do servidor RJSRVUXX02 – Estratégia LVM Split Mirror

Conforme visto na **Figura 3**, responsável por consolidar o mapeamento realizado no servidor RJSRVUXX02, esse host tem apenas um LVOL (Logical Volumes) de 80 GB para ser migrado atualmente hospedado em um único PV (Physical Volume).

Nesse procedimento, utilizaremos uma técnica muito interessante para migrar nossos dados, conforme observado na



FileSystem	Device	Tamanho (Gb)	FS	Volume Manager	VG	LV	PE	PE Size
/srv/app	/dev/app_vg/lvol_01	80,00	ext3	LVM	app_vg	lvol_05	---	---

Figura 3. Dados a serem migrados do servidor RJSRVUXX02

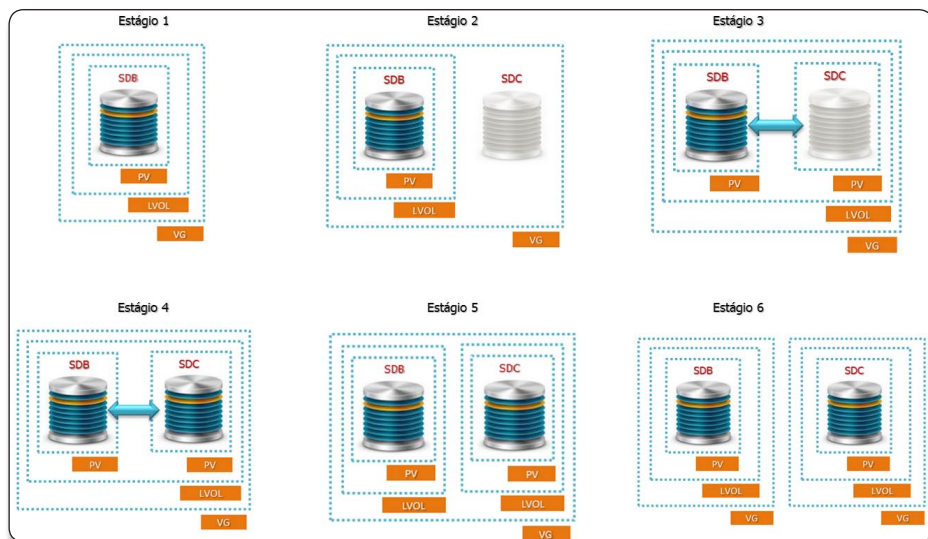


Figura 4. Estágios do procedimento "LVM – Split Mirror" executado no servidor RJSRVUXX02

Figura 4, que consiste em realizar um espelhamento de dados entre disco/LUN origem e destino. Depois do espelhamento estabelecido e todos os dados copiados entre disco/LUN origem e destino, realizaremos o *SPLIT* (quebra) do espelhamento entre os dois discos/LUNs. Após a execução desse procedimento, ao final do artigo teremos dois *Volume Groups* compostos por cada um dos dois lados do espelhamento.

Normalmente essa técnica é utilizada se desprezando o *Volume Group* que contém os discos originais imediatamente após o término do processo de espelhamento, porém, para propósitos de *Rollback*, é interessante manter ambas as estruturas LVM antiga e nova até que se tenha certeza de qual dos lados será utilizado ao término do processo de migração dos dados.

Imagine uma migração de alguns *Tera-bytes* que demore cerca de 12 horas para ocorrer. Imagine também que ocorra algum problema de desempenho com o novo *Storage* de forma que os novos discos/LUNs não possam ser utilizados, e que seja

necessário realizar um *Rollback* da migração dos dados depois de a migração ter sido concluída com sucesso. Uma janela de manutenção de 12 horas pode se transformar em uma janela de 24 horas, pois para realizar o *Rollback* os dados precisam ser sincronizados novamente.

Para evitar a necessidade de nova sincronização do espelhamento em caso de *Rollback* após o término do processo, ou seja, duplicando a janela de manutenção, esse artigo vai focar em manter os dois lados do espelhamento depois do término do processo de migração dos dados. A **Listagem 2** consolida todos os procedimentos operacionais que devem ser adotados para migrar com sucesso os dados do servidor RJSRVUXX02.

Para facilitar a compreensão da sequência de comandos exibida na **Listagem 2**, a dividimos em várias partes, e agora explicaremos cada uma separadamente:

1. Inicie o processo adicionando no servidor RJSRVUXX02 o novo disco que receberá os dados oriundos do processo de migração. Em nosso exemplo, o novo

disco já foi reconhecido no sistema operacional com o caminho `/dev/sdc` (2:0:0:0).

Depois de o novo disco já estar reconhecido no sistema operacional, preencha a coluna “New Dev” da tabela “De/Para”, inserindo o caminho do novo disco/LUN reconhecido no passo anterior conforme observado na **Tabela 2** e destacado em negrito;

HOST	VG	Old Dev	New Dev	PE Size	PE	SIZE
RJOSRVUXX02	app_vg	/dev/sdb1	/dev/sdc1	4 M	20478	80 GB

Tabela 2. Tabela De/Para preenchida do servidor RJOSRVUXX02

2. Entre no utilitário FDISK e crie uma partição de tamanho igual ou superior a `/dev/sdb1` no disco `/dev/sdc`. Adicionalmente, não se esqueça de colocar o tipo da partição como “8E”; caso contrário, não será possível executar os próximos passos do tutorial e não será possível criar um PV na partição física do novo disco;
3. Visualize a estrutura LVM que será migrada conforme o Estágio 1 da **Figura 4**. Para facilitar a compreensão do procedimento que será executado, o Logical Volume `lvol_01` existente no device `/dev/sdb1` será espelhado para o device `/dev/sdbc1`.

Antes que se possa movimentar o PV, primeiramente devemos criar o Physical Volume na partição que preparamos no passo 2 (`/dev/sdc1`);

4. Depois do Physical Volume criado no passo anterior, agora adicione o PV `/dev/sdc1` ao Volume Group `app_vg` utilizando o comando `vgextend`. Nesse instante, a estrutura LVM deve estar semelhante à estrutura visualizada no Estágio 2 da **Figura 4**;
5. A partir desse ponto, a estrutura LVM já pode ser espelhada com sucesso. Utilize o comando `lvconvert` e inicie a conversão do Logical Volume `lvol_01` para o formato espelhamento (RAID 1), utilizando o device `/dev/sdc1` como disco que receberá o novo espelhamento.

A opção “-m1” informa um espelhamento e a opção “-corelog” informa que o log de consistência entre o `submirror 1` e o `submirror 2` será mantido em memória em vez de ser criado em disco. A utilização da opção “-corelog” facilitará o processo de `SPLIT` do espelhamento que será executado mais adiante.

A velocidade de cópia varia de acordo com diversos fatores, como velocidade da controladora do host e barramento que acessa os dados, concorrência com os demais I/Os destinados à controladora do Storage por outros hosts, taxa de acesso da aplicação, entre outros fatores.

Listagem 2. Sequência de comandos para migração do host RJOSRVUXX02.

```
##### Trecho 1 #####
sysadmin@rjosrvuux02:~$ sudo lsccsi --size
[0:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sda 8.58GB
[1:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sdb 85.8GB
[2:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sdc 85.8GB

##### Trecho 2 #####
sysadmin@rjosrvuux02:~$ sudo fdisk /dev/sdc

Command (m for help): n
Partition type:
  p primary (0 primary, 0 extended, 4 free)
  e extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-167772159, default 2048): <enter>
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-167772159, default 167772159): <enter>
Using default value 167772159

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 8e

Command (m for help): w
The partition table has been altered!

##### Trecho 3 #####
sysadmin@rjosrvuux02:~$ sudo pvcreate /dev/sdc1
Physical volume "/dev/sdc1" successfully created

##### Trecho 4 #####
sysadmin@rjosrvuux02:~$ sudo vgextend app_vg /dev/sdc1
Volume group "app_vg" successfully extended

##### Trecho 5 #####
sysadmin@rjosrvuux02:~$ sudo lvconvert -m1 --corelog app_vg/lvol_01 /dev/sdc1
app_vg/lvol_01: Converted: 0.0%
app_vg/lvol_01: Converted: 22.3%
app_vg/lvol_01: Converted: 54.7%
app_vg/lvol_01: Converted: 72.9%
app_vg/lvol_01: Converted: 99.3%
app_vg/lvol_01: Converted: 100.0%

##### Trecho 6 #####
sysadmin@rjosrvuux02:~$ sudo lvs
LV VG Attr LSize Origin Snap% Move Log Copy% Convert
lvol_01 app_vg mwi-ao 79.90g 100.00

sysadmin@rjosrvuux02:~$ sudo lvs -a -o+devices

##### Trecho 7 #####
sysadmin@rjosrvuux02:~$ sudo umount /srv/app

##### Trecho 8 #####
sysadmin@rjosrvuux02:~$ sudo lvconvert --splitmirrors 1 -n new_lv01 app_vg/
lv01 /dev/sdb1
Logical volume lv01 converted.

##### Trecho 9 #####
sysadmin@rjosrvuux02:~$ sudo lvs
LV VG Attr LSize Origin Snap% Move Log Copy% Convert
lvol_01 app_vg -wi-a- 79.90g
new_lv01 app_vg -wi-a- 79.90g

sysadmin@rjosrvuux02:~$ sudo mount /srv/app

##### Trecho 10 #####
sysadmin@rjosrvuux02:~$ lvchange -an app_vg/new_lv01

##### Trecho 11 #####
sysadmin@rjosrvuux02:~$ vgsplit app_vg app_vg_old /dev/sdb1
```

A aplicação ou serviço que utiliza os dados que estão sendo migrados não precisa estar em estado OFFLINE nesse momento. O espelhamento pode ocorrer com a aplicação ONLINE em estado produtivo.

Durante o andamento do processo de espelhamento, a estrutura LVM será idêntica à vista no Estágio 3 da **Figura 4**;

6. O andamento do espelhamento pode ser observado com o comando `lvs` na coluna *Copy %*, conforme destacado em negrito no output do comando.

Em nosso caso, o espelhamento já se encontra finalizado com status de sincronismo de 100% dos dados. Esse já é o ponto seguro para realizar o SPLIT da estrutura LVM. Nessa fase, a estrutura LVM estará como ilustrada no Estágio 4 da **Figura 4**;

7. Baseado no output do comando `lvs` que mostra o estado do sincronismo (*Copy %*) em 100%, já podemos realizar com segurança o SPLIT dos dados.

Caso seja do interesse do leitor manter os dois lados do espelhamento (antigo e novo) conforme o artigo sugere, é necessário gerar um ponto de consistência da aplicação. Se ocorrer SPLIT do espelhamento com a aplicação ativa, apenas o lado novo do espelhamento continuará sendo atualizado com os I/Os destinados ao volume. O lado antigo do espelho não será mais atualizado com as alterações que ocorrem no volume depois do momento do SPLIT, podendo inclusive estar corrompido ao final do processo e não servir para efeitos de Rollback porque o *File System* estava montado com a aplicação ativa.

Para haver consistência dos dados, em nosso exemplo vamos efetuar UMount do *File System* `/srv/app` para que cada um dos lados do espelho tenha uma imagem idêntica e íntegra. Caso qualquer um dos lados do espelhamento venha a ser utilizado no futuro, devido à operação de SPLIT ter sido executada com o *File System* desmontado, não será necessária a execução da ferramenta FSCK (verificação de integridade).

Existem algumas aplicações comerciais destinadas ao universo Datacenter que oferecem recursos adicionais de consistência de dados para realizar o processo de SPLIT de espelhamento com integridade dos dados, evitando a necessidade de parada da aplicação

ou `umount` do *File System*. O banco de dados Oracle, por exemplo, tem um recurso que permite colocar todas as suas TableSpaces (estruturas internas que armazenam as informações) em modo Backup (BEGIN BACKUP). Dessa forma, pode ocorrer SPLIT do espelhamento de forma íntegra. Após o término do processo, as TableSpaces precisam ser tiradas do modo Backup.

Por meio do comando `umount`, efetue a desmontagem do *File System* para gerar o ponto de consistência;

8. Mediante geração do ponto de consistência da aplicação, utilize o comando `lvconvert` para efetuar o SPLIT do espelhamento. Durante este processo, o administrador precisa informar o nome do novo Logical Volume, de forma que ao final dessa etapa tenhamos dois Logical Volumes dentro de um mesmo Volume Group, conforme o Estágio 5 da **Figura 4**.

A opção “-n new_lv01” define o nome do novo Logical Volume, em nosso caso, “new_lv01”.

O comando `lvconvert` também precisa receber como parâmetro o nome do volume que sofrerá a operação de SPLIT, em nosso caso, “app_vg/lvol_01”, e por fim qual o disco que vai permanecer associado ao Logical Volume antigo, em nosso caso, “/dev/sdb1”;

9. Depois de ocorrer o SPLIT do espelhamento, verifique com o comando `lvs` que a partir de agora existem dois Logical Volumes, `lv01_01` e `new_lv01_01`, com aproximadamente 80 GB cada um, e que o Volume antigo pode voltar a ser acessado sem problema algum, com os dados íntegros refletindo o momento prévio ao SPLIT. Monte novamente o *File System*;

10. No momento atual temos dois Logical Volumes dentro de um único Volume Group. Agora, para melhorar a gerenciabilidade dos dados, realizaremos a operações

de SPLIT do Volume Group de forma que, ao final do processo, tenhamos dois Volume Groups com um Logical Volume cada, conforme pode ser observado no Estágio 6 da **Figura 4**. A aplicação pode estar ONLINE nesse momento, pois apenas o novo volume (`new_lv01_01`) que será desmembrado do Volume Group `app_vg` será afetado com a operação de Split do Volume Group.

Garanta a consistência dos dados deixando o volume `new_lv01_01` em estado desabilitado com o comando `lvchange` antes de realizar o Split do Volume Group;

11. Utilize o comando `vgsplit` para realizar o SPLIT do Volume Group `app_vg`. O comando `vgsplit` precisa receber como parâmetros o nome do Volume Group que sofrerá operação de SPLIT (`app_vg`), o nome do novo Volume Group que será criado (`app_vg_old`) depois da operação e quais Physical Volumes irão para esse novo Volume Group; em nosso caso, `/dev/sdb1`.

Cuidado com a execução do comando `vgsplit`. Na dúvida, consulte as MAN PAGES do comando para ter certeza da sintaxe que você está executando. Em nosso caso, a utilização da nomenclatura `app_vg_old` se deve ao fato de o novo Volume Group que será criado ser composto do disco antigo (`/dev/sdb`) que já existia antes de espelhamos o volume. Ao final dessa etapa, o Volume Group antigo (`app_vg`) conterá o novo disco (`/dev/sdc`).

Migração do servidor RJOSRVUXX03 – Estratégia RSYNC

Diferentemente dos demais servidores que até então passaram pelos procedimentos operacionais de migração de dados propostos pelo artigo, o servidor RJOSRVUXX03, conforme visto na **Figura 5**, não tem seus dados gerenciados pelo LVM. O mesmo possui apenas um *File System* (`/srv/monitor`) de 80 Gb montado atualmente como Hard Partition.

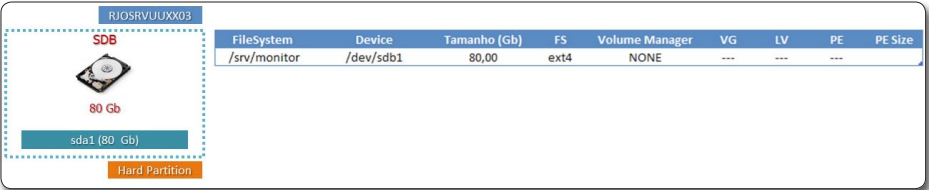


Figura 5. Dados a serem migrados do servidor RJOSRVUXX03

Já vimos também em outros momentos desse artigo que a situação de *Hard Partition (/dev/sdb1)* não é interessante para migração de dados ONLINE, pois é necessário parar a aplicação para gerar um ponto de consistência dos dados que serão migrados. Não podemos, por exemplo, utilizar o espelhamento via sistema operacional quando os dados não estão sob gestão do LVM.

Nesse procedimento, utilizaremos a ferramenta *RSYNC* nativa do próprio sistema operacional Linux para efetuar a cópia entre o *File System* antigo e o novo. O utilitário *RSYNC* possui recursos muito interessantes para esse tipo de migração, oferecendo recursos como preservar *Hard Links*, *Soft Links*, controle de banda de rede, possibilidade de controle de integridade baseado em *Checksum*, entre outras voltadas às necessidades mais comuns dos ambientes Datacenter.

A ferramenta *RSYNC* também suporta a sincronização entre servidores remotos utilizando SSH. Nesse cenário basta informar como origem ou destino da cópia um caminho no formato “usuário@servidor:/caminho/completo” e a transferência ocorrerá de forma criptografada. Em nosso tutorial, não efetuaremos uma cópia remota, mas sim local, no próprio servidor, entre um *File System* origem e outro *File System* destino.

Outra vantagem interessante do utilitário *RSYNC* é a possibilidade de execução de cópias incrementais dos dados. É possível executar várias sessões de *RSYNC* de forma que se possa copiar

os dados aos poucos, em operações incrementais. Sempre que uma nova sessão é estabelecida, todos os arquivos são verificados novamente, de forma que aqueles que já estiverem copiados e idênticos entre origem e destino não sejam copiados novamente.

Para facilitar a compreensão da sequência de comandos exibida na **Listagem 3**, a dividimos em várias partes e agora explicaremos cada uma separadamente:

1. Inicie o processo adicionando no servidor *RJOSRVUXX03* o novo disco que receberá os dados oriundos do processo de migração. Em nosso exemplo o novo disco já foi reconhecido no sistema operacional com o caminho */dev/sdc* (2:0:0:0).

Depois de o novo disco já estar reconhecido no sistema operacional, preencha a coluna “New Dev” inserindo o caminho do novo disco/LUN reconhecido no passo anterior, conforme destacado em negrito na **Tabela 3**;

HOST	VG	Old Dev	New Dev	PE Size	PE	SIZE
RJOSRVUXX03	---	/dev/sdb1	/dev/sdc1	---	---	80 GB

Tabela 3. Tabela De/Para preenchida do servidor *RJOSRVUXX03*

2. Entre no utilitário *FDISK* e crie uma partição de tamanho igual ou superior a */dev/sdb1* no disco */dev/sdc*. Também é recomendado manter o tipo da partição como “83”, diferente do formato LVM que era “8E”;

Listagem 3. Sequência de comandos para migração do host *RJOSRVUXX03*.

```
##### Trecho 1 #####
sysadmin@rjosrvuux03:~$ sudo lsscsi --size
[0:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sda 8.58GB
[1:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sdb 85.8GB
[2:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sdc 85.8GB

##### Trecho 2 #####
sysadmin@rjosrvuux03:~$ sudo fdisk /dev/sdc

Command (m for help): n
Partition type:
 p primary (0 primary, 0 extended, 4 free)
 e extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-167772159, default 2048): <enter>
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-167772159, default 167772159): <enter>
Using default value 167772159

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 83

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

##### Trecho 3 #####
sysadmin@rjosrvuux03:~$ sudo mkfs.ext3 /dev/sdc1

##### Trecho 4 #####
$ sysadmin@rjosrvuux03:~$ sudo mount -t ext3 /dev/sdc1 /srv/new_monitor

##### Trecho 5 #####
sysadmin@rjosrvuux03:~$ df -h | tail -2
/dev/sdb1      79G  40G  39G  50% /srv/monitor
/dev/sdc1      79G 184M  75G   1% /srv/new_monitor

##### Trecho 6 #####
sysadmin@rjosrvuux03:~$ sudo rsync -axvH /srv/monitor/ /srv/new_monitor/

##### Trecho 7 #####
sysadmin@rjosrvuux03:~$ cd /srv/monitor
sysadmin@rjosrvuux03:~$ sudo tar xf - | ( cd /srv/new_monitor ; tar xf - )

##### Trecho 8 #####
sysadmin@rjosrvuux03:~$ sudo rsync -axvH /srv/monitor/ /srv/new_monitor/
sending incremental file list

sent 22256 bytes received 202 bytes 44916.00 bytes/sec
total size is 47329209 speedup is 2107.45

##### Trecho 9 #####
sysadmin@rjosrvuux03:~$ sudo rsync -acxvH /srv/monitor/ /srv/new_monitor/

##### Trecho 10 #####
sysadmin@rjosrvuux03:~$ sudo umount /srv/monitor
sysadmin@rjosrvuux03:~$ sudo umount /dev/sdc1 /srv/monitor
```

3. Para que se possa utilizar a nova área em disco existente no device `/dev/sdc`, é necessário criar um *File System*. Baseado no levantamento de informações que foi realizada no início do artigo, também foi documentado que o *File System* atual (`/srv/monitor`) é do tipo `ext3`;

4. Efetue a operação de montagem do novo *File System* em algum *Mount Point* temporário, em nosso caso, `/srv/new_monitor`;

5. Execute o comando `df -h` e verifique se o *File System* origem e destino já estão relacionados e disponíveis. Nos próximos passos, copiaremos todo o conteúdo do *File System* `/srv/monitor` para dentro do *File System* `/srv/new_monitor`;

6. Estabeleça um ponto de consistência dos dados efetuando o `STOP` das aplicações e serviços que utilizam o *File System* `/srv/monitor` que será migrado em seguida. Caso a aplicação não seja interrompida, os dados que estão sendo migrados não estarão confiáveis, podendo haver arquivos corrompidos.

Efetue a cópia dos dados utilizando a ferramenta `RSYNC`. O `RSYNC` pode receber várias opções de execução que controlam a forma que a ferramenta trabalha. Em nosso caso, são utilizadas as opções “-axvH”. Em seguida, o `RSYNC` precisa receber o caminho origem da cópia (`/srv/monitor/`), e, por fim, o caminho destino que receberá os dados (`/srv/new_monitor`).

A utilização das opções “-axvH” do utilitário `RSYNC` é recomendada para manter a integridade da estrutura de arquivos e diretórios que será migrada. Com a opção “-v” estamos habilitando o modo *Verbose*, a fim de mostrar mais informações durante a cópia. A opção “-x” orienta o `RSYNC` a não ultrapassar *Mount Points* caso um *File System* tenha *Mount Point* dentro de outro. A opção “-H” preserva a existência de *Hard Links*, caso existam. Já a opção “-a” é uma combinação de várias outras opções que ajuda a preservar todos os atributos originais do arquivo como características de permissão, *owner/group* (propriedade) e tipo de arquivo;

7. O comando `TAR` ou outra ferramenta de sua preferência também poderia ser utilizada para migrações de dados, porém cada uma tem sua limitação. O utilitário `TAR`, por exemplo, não permite arquivamento de arquivos maiores de 4GB em alguns sistemas operacionais UNIX mais antigos. Em outros casos, é necessário usar diversas opções do comando para manter os devidos permissionamentos e a integridade da estrutura de arquivos, diretórios e links. Por meio do comando `TAR`, realize a migração dos dados;

8. Depois de a primeira execução do comando `RSYNC` ter finalizado com sucesso, é saudável verificar que todos os dados permanecem íntegros e que nada mudou em relação ao momento de início da primeira execução do `RSYNC`. Dessa forma, execute novamente o comando `RSYNC`.

Não se preocupe em executar o `RSYNC` uma segunda ou terceira vez, pois os arquivos não serão copiados novamente. A execução de várias sessões é uma prática comum quando o `RSYNC` é utilizado. A ferramenta `RSYNC` possui um recurso de verificação entre caminho origem e destino. Os arquivos somente são copiados novamente se algum atributo for alterado, como permissão, proprietário, grupos, tamanho e conteúdo. Se alguma dessas

situações ocorrer, o próprio `RSYNC` copia o arquivo ou diretório novamente. Caso contrário, nenhuma nova transferência é feita, apenas a verificação de integridade entre origem e destino;

9. Conforme explicado anteriormente, quando o mesmo arquivo já existe na origem e no destino, o mecanismo de verificação de integridade do `RSYNC` por padrão se baseia em Metadados (permissão, atributos e tamanho) do arquivo para entender se alguns desses itens foram alterados e se é necessário copiar novamente o arquivo. O utilitário `RSYNC` possui uma opção “-c” que troca esse método de validação de consistência por uma verificação de consistência baseada no *Checksum* do conteúdo do arquivo. Nesse caso, não só as informações de Metadados são comparadas entre origem e destino para decidir se um dado arquivo será copiado, mas também o conteúdo do arquivo em si.

As operações de verificação de integridade baseadas em *Checksum* devem ser utilizadas com cautela, pois demoram mais e consomem mais CPU. Isto ocorre porque além dos Metadados, o conteúdo do arquivo também precisa ser verificado;

10. Após o término do processo, efetue a desmontagem do *File System* antigo e monte o novo device utilizando o *Mount Point* anterior. Também vale ressaltar que talvez seja necessário um ajuste no arquivo `/etc/fstab` que mantém a relação dos *File Systems* que são montados automaticamente em momento do *Boot* do sistema operacional, refletindo a alteração entre o device antigo (`/dev/sdb1`) e o novo (`/dev/sdc1`).

Desse ponto em diante, os dados foram migrados com sucesso e já pode dar início à fase de validação do processo de migração dos dados.

Validação da Migração

Não menos importante que o processo de migração dos dados em si é o processo de validação da migração, que normalmente consiste em testes para evidenciar que o ambiente ou sistema encontra-se ok com todos os dados íntegros depois da movimentação dos dados.

Um processo adequado de validação da migração dos dados consiste em execuções de tarefas e rotinas da aplicação que utilizem de alguma forma os dados que foram migrados. Uma simples inicialização da aplicação em muitos casos não se mostra um procedimento adequado de validação, pois nem toda aplicação valida a integridade de seus dados no momento da inicialização.

Certas aplicações como bancos de dados voltados ao mercado *Enterprise* possuem mecanismos de consistência de dados baseados em *checksum* ou em consistência de blocos, de forma que um *start* com sucesso já pode ser suficiente em alguns casos para validar a migração dos dados. Existem outras aplicações, por sua vez, que só validam os dados mediante um acesso formal que resulte no I/O nos blocos que foram migrados. Às vezes é necessário realizar consultas, gerar relatórios ou fazer requisições específicas que validem referências de índices, estruturas de dados e algum ponto específico do sistema em questão.

Mesmo nos casos de sucesso do *start* da aplicação ou na validação adequada de consistência, ainda é importante validar aspectos de

desempenho como tempo de resposta dos novos discos/LUNs e como esse reflete na aplicação. Não são raros casos em que um ambiente é migrado e validado na janela de manutenção, porém, quando a aplicação entra em janela produtiva e recebe a carga cotidiana de seus milhões de acessos, o ambiente se comporta mal nos novos discos ou Storage e pode não haver mais tempo hábil para uma análise mais apurada e definitiva que não resulte em impacto aos negócios.

Existem casos em que o cliente adquire um modelo de Storage novo, podendo ser de outro fabricante, outra formatação física, bem como outro algoritmo de gerência dos discos, mecanismos de cache, entre outros, de forma que a performance final pode ser muito diferente da normalmente vivenciada no hardware antigo. É normal haver configurações específicas relacionadas a SAN, FC, SCSI, iSCSI e FCOE, que podem não estar com valores adequados para trabalhar com o novo equipamento, resultando em baixo desempenho ou desempenho insatisfatório.

Por isso é importante estabelecer testes apropriados de validação que ao menos simulem a quantidade ou o volume de acessos próximo do real, de modo que reflitam em informações consistentes de desempenho que possam atestar e evidenciar realmente o sucesso da migração de dados do ponto de vista de consistência e velocidade de acesso.

Adicionalmente, sempre considere a possibilidade de Rollback. Normalmente é muito comum ninguém se preocupar com isso durante o planejamento ou a definição da duração da janela adequada de manutenção. A consequência disso pode ser desastrosa quando o tempo necessário para Rollback não é incluído na janela de manutenção. Rollback não significa “incompetência” ou “não saber fazer”, como muitos consideram. Às vezes problemas não esperados ocorrem como desempenho não satisfatório ou bugs.

A estratégia de *Split Mirror* do LVM mostra-se muito interessante nesse aspecto. É possível ainda manter os volumes espelhados por um período definido, configurando qual lado do mirror vai responder preferencialmente aos I/Os. Assim, é possível manter os dados sincronizados e somente realizar o split quando houver certeza absoluta de que tudo se encontra conforme esperado e quando o resultado final for satisfatório.

Clean-Up do ambiente

Mediante confirmação do sucesso da migração e validação adequada dos dados migrados, chegou o momento de *Clean-up* do ambiente.

Em alguns casos, os dados residiam em um Storage que será desativado e necessita ter os discos/LUNs removidos do servidor. Nesse caso, além da remoção dos discos do *host*, algumas etapas como remoção do ZONE na SAN (*Storage Area Network*) e remoção de *Masking* e *Mapping* no subsistema Storage também devem ser executados.

Em outros casos, quando os *Mount Points* sofrem alteração, alguns outros sistemas adjacentes no Datacenter como Monitoramento, Backup, automação, entre outros, devem ser acionados

ou sinalizados da alteração que ocorreu. Também é muito comum que esse ponto seja esquecido ou não considerado no planejamento, gerando problemas para outra equipe que não aquela que realizou a manutenção no ambiente. Imagine as políticas de Backup que se baseiam em *Mount Points*. Caso esse *Mount Point* mude, o Job de Backup irá falhar. Caso a equipe que realizou a migração não seja a mesma que mantenha o ambiente de backup, a causa raiz do problema não estará clara e será desperdiçado um esforço desnecessário para resolução do problema.

Outra necessidade de alteração dos controles internos de uma equipe ocorre quando o *Mount Point* de um determinado *File System* é alterado na migração. Vale lembrar que ferramentas de monitoramento como o Nagios e OpenView podem conter scripts que se baseiam no *Mount Point* para monitorar o *File System*. Nesse caso também é necessário realizar alterações nessas ferramentas.

Outro ponto que é impactado durante os processos de migração de dados são os Frameworks de alta disponibilidade. É comum a existência de Clusters que mantêm em sua configuração o caminho absoluto dos *device files* (*/dev/sda*) que referenciam as LUNs que estão sendo migradas. Sendo assim, estes também requerem adequações em suas configurações. Caso o administrador não realize as alterações de *Mount Point* ou caminho dos discos nas configurações do Cluster, haverá problemas para iniciar a aplicação depois de realizar a migração dos dados.

As políticas de backup também costumam ser impactadas e devem ser reconfiguradas refletindo as alterações ocorridas no processo de migração. Existem soluções de Backup que utilizam o caminho dos *RAW devices* (*/dev/raw3*), por exemplo, e são sensíveis a migrações de dados que alterem o PATH de acesso ao dispositivo.

Também é comum encontrar aplicações web em ambientes Datacenter que tenham interfaces externas como áreas de upload destinadas a receber arquivos de outros clientes ou outras aplicações. É natural que essas aplicações tenham atributos ou parâmetros de configuração que se baseiem em algum *File System* ou *Mount Point*. Quando o *Mount Point* é alterado durante a migração de dados é fundamental incluir no planejamento da migração a reconfiguração desse atributo ou parâmetro refletindo as mudanças. Caso não seja realizado, clientes externos ou aplicações dependentes serão afetadas gerando um impacto negativo aos negócios.

Outro ponto pouco lembrado nas migrações de dados é a adequação das documentações de controle depois da migração ser concluída. Em centros computacionais de grande porte, existem diversas equipes e colaboradores que mantêm vasta documentação do ambiente gerenciado através de planilhas relacionando endereços IP, nomes de servidores, aplicações, caminhos de arquivos, entre outros. Equipes como a de suporte a Bancos de Dados, suporte a Servidores e Sistemas Operacionais, suporte a Infraestrutura de Network e suporte a ambientes WEB mantêm esses dados que vão se tornando obsoletos com o passar dos anos e podem dificultar a gestão ou manutenção do ambiente,

contrariando seu propósito. Portanto, é muito importante para a gestão de TI que essas informações sejam alteradas e mantidas corretamente a cada migração, permitindo que todos os controles estejam sempre bem documentados.

Conclusão

Esse artigo procurou compartilhar com os leitores a abrangência de um processo que envolve movimentação de dados corporativos, abordando algumas estratégias e metodologias que podem ser empregadas para planejamento de atividades de migração em diversos ambientes de TI que disponham de sistemas operacionais Linux.

As técnicas apresentadas nesse artigo poderão ser empregadas em seu ambiente de TI, melhorando o processo atual ou servindo de parâmetro para implementação de um novo processo de mudanças que envolvem tarefas de migração de dados.

A primeira parte deste artigo, apresentada na edição anterior da Infra Magazine, procurou abordar aspectos conceituais com propósitos de planejamento do processo de migração de dados corporativos. Adicionalmente, na primeira parte, todo o ambiente utilizado em nossos testes foi mapeado por meio da execução de comandos específicos nos três servidores propostos.

Já a segunda parte do artigo, objeto desta edição, procurou compartilhar com os leitores os procedimentos técnicos utilizados para realizar a migração de dados em si. As estratégias foram traçadas baseadas em metodologias de mercado respeitando as diretrizes operacionais do negócio. Para escolher qual estratégia utilizar para movimentar os dados, cada um dos três servidores

passou por um processo de análise considerando a condição em que o dado encontrava-se hospedado. Neste cenário, para melhor didática e compreensão dos leitores, procuramos abordar técnicas para movimentar os dados utilizando estratégias de LVM ou empregando ferramentas nativas do sistema operacional Linux como RSYNC ou TAR, presentes em qualquer servidor de ambiente Datacenter.

O foco desse artigo foi discutir e apresentar estratégias de migrações de dados por meio do sistema operacional Linux ou gerenciador de volumes LVM. No entanto, existem inúmeras outras técnicas e modalidades que também podem ser empregadas para migrar dados, como estratégias RAID, manobras com *File Systems* NFS, produtos específicos oferecidos por fabricantes de Storages, Appliances, estratégias LVM adicionais, entre outras.

Autor



Telmo Eduardo Nóbrega Reis

telmoeduardo@gmail.com

Analista de sistemas e consultor especialista em Storage, SAN e sistemas operacionais UNIX/Linux. Atualmente trabalha como arquiteto de soluções em projetos que envolvam tecnologias de alta disponibilidade, migração e replicação de dados, Capacity & Planning, análise e reestruturação de Infraestrutura LAN, WAN e SAN, Continuidade de Negócio, Disaster & Recovery e Cloud Computing. Atuou em projetos importantes em companhias como Banco Votorantim, VIVO, TIM, Nextel, Tivit, Diveo, HP, EMC, Symantec, Sun Microsystems, BM&F Bovespa, Embraer, Huawei, entre outras. Iniciou sua jornada no mercado de TI na companhia Sun Microsystems (agora Oracle) onde atuou como instrutor certificado por 5 anos lecionando sistema operacional Solaris, SunCluster e produtos relacionados a Storage.



RENOVE JÁ!

Sua assinatura pode estar acabando

Renovando a assinatura de sua revista favorita você ganha brindes e descontos exclusivos.



www.devmedia.com.br/renovacao ou (21)3382-5038



Somos tão apaixonados por tecnologia que o nome da empresa diz tudo.

Porta 80 é o melhor que a Internet
pode oferecer para sua empresa.

Já completamos 8 anos e
estamos a caminho dos 80, junto
com nossos clientes.

Adoramos tecnologia.
Somos uma equipe composta
de gente que entende e
gosta do que faz,
assim como você.



Estrutura

100% NACIONAL.
Servidores de primeira
linha, links de alta
capacidade.



Suporte diferenciado

Treinamos nossa equipe
para fazer mais e melhor.
Muito além do esperado.



Serviços

Oferecemos a tecnologia
mais moderna, serviços
diferenciados e
antenados com as suas
necessidades.



1-to-1

Conhecemos nossos
clientes. Atendemos
cada necessidade de
forma única.
Conheça!



Porta 80
WEB HOSTING

Hospedagem | Cloud Computing | Dedicados | VoIP | Ecommerce |
Aplicações | Streaming | Email corporativo

porta80.com.br | comercial@porta80.com.br | twitter.com/porta80

SP 4063-8616 | RJ 4063-5092 | MG 4063-8120 | DF 4063-7486