Vinny Moreira - www.sxc.hu

Usando o *rsync*

Sincronia total

É muito bom descobrir que ainda existem programas capazes de nos impressionar. Aqueles que, após o uso, nos deixam balançando a cabeça pra cima e pra baixo, com os lábios franzidos, estupefatos. O *rsync* é um desses programas. POR JULIANO SIMÕES E MARCELO BARROS DE ALMEIDA

O *rsync* foi escrito por Andrew Tridgell (sim, o mesmo autor do consagrado *Samba*) e Paul Mackerras, baseado no algoritmo de mesmo nome que ambos desenvolveram em 1996. Atualmente o programa é mantido por Martin Pool. O projeto tinha como objetivo, desde o princípio, otimizar ao máximo a transferência de arquivos entre uma máquina remota e outra local, valendo-se das estratégias mostradas a seguir:

- Somente são transmitidas as diferenças entre arquivos nas duas máquinas, evitando-se o envio de arquivos que não foram alterados ou que já existem "do outro lado".
- Mesmo para arquivos modificados, somente as partes diferentes são transferidas. Para isto, o rsync divide o arquivo em blocos, calcula e transmite um *checksum* (soma de verificação) para cada um destes blocos. Cada "lado" verifica os checksums recebidos e transmite apenas os blocos com checksums diferentes.
- Cada bloco enviado pode ser compactado usando o mesmo algoritmo do consagrado gzip.
- O canal de comunicação é totalmente aproveitado: cada lado possui dois processos rodando independentemente. Enquanto um gera e envia os checksums, o outro recebe os checksums e reconstrói os arquivos.

O *rsync* ainda pode fazer toda esta transferência utilizando uma conexão segura *(ssh)*, com acesso autenticado ou anônimo, preservar todos os direitos e permissões dos arquivos, copiar links simbólicos, gerar listas de exclusão (arquivos que não devem ser transferidos), limitar o uso da banda e rodar sem privilégios de administrador.

Com tantas qualidades, não é difícil entender porque o *rsync* é cada vez mais usado para espelhamento (mirror) e backup de dados. Por exemplo, os vários mirrors do Samba são mantidos em sincronismo através do *rsync*. O mesmo vale para alguns mirrors da Debian, que já há um bom tempo usam o *rsync* para manter as imagens ISO dos CDs que compõem a distribuição. A parte pública do servidor ftp.kernel.org também pode ser acessada via *rsync*.

Modos de utilização e principais opções

Basicamente, uma operação do *rsync* envolve a especificação de opções (*OPTIONS*) e de dois caminhos: um fonte (*SRC*) e um destino (*DST*) para a transferência, da seguinte forma:

```
rsync OPTIONS SRC DST
```

Pelo menos um dos caminhos tem que ser local. O outro pode ou não ser remoto. Quando um deles é remoto (indicado abaixo por *HOST*) e um shell é usado como transporte para a transferência (ssh ou rsh, por exemplo), a notação passa a incluir o caracter ":", como mostrado a seguir:

```
rsync OPTIONS HOST:SRC DEST
rsync OPTIONS SRC HOST:DEST
```

O uso do shell ssh como transporte é bastante interessante, gerando uma forma segura de transferência de dados.

Outra opção é acessar um servidor rsync (geralmente instalado na porta 873) diretamente, sem passar por um shell. Nessa situação não se tem a segurança no transporte dos dados fornecida pelo ssh, já que o próprio *rsync* transfere os dados. A string "::" é usada para indicar esse tipo de acesso.

Nessa situação, o caminho SRC está relacionado a um módulo e não mais a um caminho absoluto. Um módulo é definido como um nome simbólico que representa uma determinada parte do disco. Por exemplo, o nome simbólico *cvs* poderia ser definido como um módulo relacionado ao caminho */var/cvsroot*, compartilhando tudo que se encontra a partir desse diretório. O uso de módulos permite um melhor gerenciamento das áreas exportadas pelo rsync, evitando que o cliente tenha que saber a real posição dos arquivos. Também permite mais segurança, já que se

tem um controle mais fino dos módulos. A forma genérica de acesso é dada por:

```
rsync OPTIONS  HOST::SRC DST
```

ou então:

```
rsync OPTIONS  SRC HOST::DST
```

A combinação "servidor de rsync + shell" também pode ser feita através da opção *--rsh* ou *-e* (só válido para a versão 2.5.6 ou posterior). É um meio bastante adequado para se ter privacidade na comunicação com um servidor *rsync*:

```
rsync OPTIONS --rsh="ssh" ➥
HOST::SRC DST
```

ou então:

```
rsync OPTIONS --rsh="ssh" SRC ➥
HOST::DST
```

Finalmente, para completar os modos de operação descritos na página de manual, o *rsync* pode ser usado para listar os módulos disponibilizados em uma máquina remota, com o comando:

```
rsync HOST::
```

ou os aquivos pertencentes a um módulo:

```
rsync HOST::SRC
rsync HOST:SRC
```

Sim, é um pouco confuso, principalmente no início. Alguns exemplos irão tornar mais claro o uso do *rsync*.

Copiando arquivos localmente

Mesmo usado localmente, o *rsync* pode ser uma ferramenta interessante, principalmente por não transferir arquivos que já estão no diretório de destino. O seguinte exemplo copia recursivamente (opção *-a*) o conteúdo do diretório */home/barros/kernel24/* para dentro do diretório */usr/local/src/linux/*, indicando o progresso da operação (opção *--progress*) e informações detalhadas sobre a transferência (opção *-v*):

```
rsync -av --progress /home/➥
barros/kernel24/ /usr/local➥
/src/linux/
```

Na realidade, a opção *-a* é equivalente a *-rlptgoD*, isto é: cópia recursiva (*-r*), preservando os links simbólicos (*-l*) e dispositivos (*-D*) no destino, mantendo as informações de permissão (*-p*), grupo (*-g*), proprietário dos arquivos (*-o*) e data de modificação (*-t*). Em resumo: uma cópia fiel dos dados originais.

Copiando arquivos remotamente

Vamos nos basear no mesmo exemplo anterior, mas agora supondo que os arquivos devam ser colocados na máquina remota dns.smar.com.br:

```
rsync -avzP /home/barros/➥
kernel24/dns.smar.com.br:/usr/➥
local/src/linux/
```

O *rsync* irá procurar por um shell para fazer este transporte e a senha será solicitada. Uma boa dica é deixar somente o ssh habilitado. Note que foram introduzidas as opções *-z* e *-P*. Com a opção *-z*, os dados são compactados antes de ser transferidos, gerando uma economia de banda. Já *-P* é equivalente a colocar as opções *--progress* e *--partial*. Esta última instrui o *rsync* a não apagar arquivos transferidos parcialmente em caso de alguma interrupção inesperada.

Sincronizando dois diretórios

Suponha que você chegou em casa e deseja sincronizar o diretório */home/barros*, na sua máquina de trabalho, com o diretório */home/barros/smar/*, no seu notebook, tornando o conteúdo do último igual ao do primeiro mas evitando os arquivos com extensão ".iso". Digite:

```
rsync -avzP --delete ➥
--exclude="*.iso" barros.smar.➥
com.br:/home/barros /home➥
/barros/smar/
```

Além da cópia dos arquivos novos e a atualização dos já existentes, a opção *--delete* apaga do destino qualquer arquivo que não esteja na origem. Se substituída pela opção *--delete-after*, também realiza o apagamento, mas só após a transferência do arquivo mais novo. Um detalhe importante: quando o diretório é especificado com a barra (*/*) no final, o conteúdo do diretório é copiado. Quando é usado sem a barra, o diretório é copiado.

Quadro 1 – rsyncd.conf

```
# rsyncd.conf - Exemplo:
#
# Configurações globais
#
# Por segurança, faz chroot para
# o caminho especificado no
# módulo abaixo
use chroot = yes
# Limita o número máximo de
# conexões simultâneas
max connections = 50
# Define as opções de log
syslog facility = local5
# Define o arquivo que conterá o
# PID do rsync
pid file = /var/bin/rsyncd.pid
# Desabilita a compressão de
# arquivos para evitar impacto
# no desempenho
dont compress = *
# Define os IPs dos clientes
# autorizados a se conectar
# ao servidor
hosts allow = 192.168.0.0/255.255.255.0
# Define o diretório /backup para
# receber/enviar os diretórios
# e arquivos dos clientes
[backup]
        path = /backup
        read only = no
```

Ou seja, neste último exemplo, existirá um diretório "barros" dentro do */home/barros/smar/*, e não somente seu conteúdo. Uma boa sugestão é usar a opção update (*-u*) para que arquivos mais novos no destino não sejam substituídos.

Acessando módulos

Para fazer o acesso a módulos em servidores *rsync*, deve ser usada a notação com a string *::*. Por exemplo, suponha que o módulo cvs esteja associado ao diretório */var/cvsroot*, na máquina 10.0.0.10. O seu conteúdo pode ser copiado para o diretório corrente da seguinte forma:

```
rsync -avz 10.0.0.10::cvs .
```

Vale notar que os diretórios dentro dos módulos podem ser acessados usando-se o nome do módulo como referência. Se houver um projeto *gnu* em */var/cvsroot/gnu*, o seu conteúdo pode ser copiado com o seguinte comando:

```
rsync -avz 10.0.0.10::cvs/gnu .
```

Listando módulos e arquivos

Para obter uma lista dos módulos disponíveis, basta o seguinte comando:

```
rsync 10.0.0.10::
```

Já a lista dos arquivos contidos no diretório */home/barros*, na máquina barros.smar.com.br pode ser obtida com o comando:

```
rsync barros.smar.com.br:/home/barros
```

Existem ainda várias opções interessantes para o *rsync* que podem ser exploradas. Uma lida na página de manual (*man rsync*) é sempre uma boa idéia.

Configurando o rsync como servidor

A fim de permitir a sincronização de arquivos entre duas máquinas, o *rsync* deve ser executado no modo servidor em pelo menos uma das pontas. O website do *rsync* possui links para download do código fonte do programa, além de binários compilados para diferentes distribuições de Linux e Unix.

Instalação no Linux/Unix

A instalação em ambientes Linux/Unix é geralmente simples, ainda mais quando feita a partir de binários .deb ou .rpm, uma vez que o *rsync* é normalmente incluído em todas as distribuições. Finalizada a fase de instalação, é necessário definir a forma de inicialização do servidor rsync que receberá conexões na porta TCP 873. No Linux existem duas opções: via inetd e em modo stand-alone. Para rodar o *rsync* em modo stand-alone, basta inserir o comando "*rsync --daemon*" em um script de inicialização do sistema (a instalação via pacote RPM já faz isto). Caso os arquivos de configuração do rsync não estejam no diretório */etc*, use *rsync --daemon --config-file=/local/do/arquivo/rsyncd.conf*

Para iniciar o daemon do rsync via inetd, é necessário incluir a seguinte linha no arquivo */etc/services*, caso ela ainda não exista:

```
rsync           873/tcp
```

e adicionar uma linha ao arquivo */etc/inetd.conf* contendo:

```
rsync   stream  tcp     nowait  root    /usr/bin/rsync rsyncd --daemon
```

Lembre-se de substituir o caminho */usr/bin/rsync* pelo local de instalação do *rsync* em seu sistema e de especificar a opção *--config-file <arquivo>* para um arquivo de configuração em local não padrão. A configuração muda um pouco para quem usa o *xinetd*, mas as adaptações são relativamente simples de fazer. Além disso, é preciso incluir as entradas referentes aos clientes que podem usar o serviço no arquivo */etc/hosts.allow*:

```
rsyncd: lista_de_IPs
```

Na lista acima, os endereços podem ser informados no formato IP/MÁSCARA DE REDE, sendo que entradas múltiplas devem ser separadas por espaços em branco.

Instalação no Windows

Quem deseja usar o *rsync* como servidor na plataforma Windows terá um pouco mais de trabalho, principalmente se decidir compilar o programa. Nesse caso, a melhor opção é usar uma versão

Quadro 2

Arquivo */etc/rsyncd.conf*

```
auth users    = root
secrets file  = /etc/rsyncd.sec
read only     = yes
uid           = root
gid           = root
hosts allow   = 10.0.0.0/24
[CVS]
  path    = /var/cvsroot
  comment = arquivos do cvs
[HTTP]
  path    = /var/www
  comment = pagina web
```

Arquivo */etc/rsyncd.sec*

```
root:senhalin
```

Quadro 3

Arquivo *C:\rsync\rsyncd.conf*

```
read only     = yes
secrets file  = c:\rsync\rsyncd.sec
read only     = yes
strict modes = false
use chroot    = false
hosts allow   = 10.0.0.0/24
[VISUALC]
  path    = c:\projetos\vc
  comment = projetos em VC++
[SRCSAFE]
  path    = d:\sourcesafe
  comment = controle de versoes
```

Arquivo *C:\rsync\rsyncd.sec*

```
root:senhawin
```

Quadro 4

Arquivo */etc/grab/lin.pass*

```
senhalin
```

Arquivo */etc/grab/win.pass*

```
senhawin
```

Quadro 5 – grab.conf

```
# grab.conf: lista de servidores
# de rsync e módulos
#
# Liste um servidor por linha,
# seguido dos módulos e do
# sistema operacional utilizado.
# Exemplo:
#
# servidor:"mod1 mod2 ... modn":so
#
# onde
# servidor: endereço do servidor
# (IP ou nome)
# mod     : módulos a sincronizar
# so      : sistema operacional
# (linux ou windows)
#
# Não use linhas em branco.
# Comentários devem começar com
# o caracter "#" na primeira
# linha.
#
# Marcelo Barros (barros@smar.
# com.br)
# 20/Oct/2003
#
10.0.0.1:"CVS HTTP":linux
10.0.0.2:"VISUALC SRCSAFE":
windows
```

já compilada para rodar com o Cygwin [1], a API para emulação do ambiente Linux no Windows. Se preferir binários já compilados, acesse o site [2].

Para rodar uma configuração mínima do rsync no Windows 9x, instale os arquivos *rsync.exe* e *cigwin1.dll* do Cygwin em uma mesma pasta (ex: *c:\rsync*) e inclua no *autoexec.bat* os comandos:

```
@echo off
rsync.exe --daemon --config=c:\rsync\rsyncd.conf .
```

Já para rodar o rsync no Windows 2000, além dos arquivos *rsync.exe* e *cigwin1.dll*, são necessários os arquivos *instsrv.exe* e *srvany.exe*, existentes no *Resource Kit* do Windows 2000. Os passos para configuração são os seguintes:

1. Grave o arquivo *rsync.exe* no diretório *c:\rsync*

2. Grave a *cigwin1.dll* em uma pasta que esteja no PATH do sistema (ex: *c:\winnt\system32*)

3. Grave os arquivos *instsrv.exe* e *srvany.exe* em um diretório temporário (ex: *c:\install*) e execute o comando mostrado a seguir a partir de uma janela do shell (*cmd.exe*):

```
c:\install> instsrv rsync "C:\install\srvany.exe"
```

4. Crie o seguinte arquivo para atualização do registro do Windows:

```
REGEDIT4
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\rsync]
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\rsync\Parameters]
"AppDirectory"="c:\\rsync"
"Application"="c:\\rsync\\rsync.exe"
"AppParameters"="--daemon --config=rsyncd.conf"
```

Para carregar as informações no registro, salve o arquivo acima como *c:\rsync\rsyncd.reg* e execute-o via Windows Explorer (basta dar dois cliques sobre o arquivo).

5. Feito isso, é possível iniciar ou parar o serviço *rsync* através do ícone de serviços, situado em *Painel de Controle -> Ferramentas Administrativas -> Serviços*. Ou, se preferir, digite *net start rsync*, em um shell qualquer.

Configurando o servidor *rsync*

O próximo passo é configurar o arquivo *rsyncd.conf*. Esse arquivo contém as diretivas lidas pelo *rsync* quando executado em modo daemon a cada nova sessão iniciada. O arquivo de configuração é composto por parâmetros, no formato *nome = valor*, e módulos identificados por colchetes. Algumas diretivas são apenas globais, isto é, não podem ser colocadas dentro do contexto de um módulo (diretiva local), e outras podem tanto ser globais como locais. Veja um exemplo comentado no Quadro 1, que ilustra a definição de apenas um módulo. Contudo, é possível definir tantos módulos quantos forem necessários para receber ou enviar arquivos para clientes remotos.

Outra questão importante diz respeito ao usuário com o qual o *rsync* será executado. Como o script não deixa isso explícito, o

Quadro 6 – grab.defs

```
#!/bin/bash
# grab.defs: definições globais
#
# Marcelo Barros (barros@smar.com.br)
# 20/OCT/03
# Diretório onde o grab está
# instalado:
GRABDIR=/etc/grab
# Local do espelhamento
MIRRORDIR=/mirror
MIRRORLOGDIR=$MIRRORDIR/log
# Local para armazenamento do backup
# rotativo
BACKUPDIR=/backup
BACKUPLOGDIR=$BACKUPDIR/log
# Opções extras para o rsync,
# dependentes do sistema operacional
RSYNCOPTCOM="--numeric-ids --stats --delete --exclude=*.iso -P "RSYNCOPTLIN="-avz  --password-file=/etc/grab/lin.pass $RSYNCOPTCOM" RSYNCOPTWIN="-rtlv --password-file=/etc/grab/win.pass $RSYNCOPTCOM"
# Opções para o RAR
RAROPTS="a -r -ol -ow -tk -ep1 "
# Email para onde os arquivos de log
# serão enviados
MAILTO=reponsavel1@meu.dominio
MAILCP=reponsavel2@meu.dominio
# caminho para os executáveis
export PATH=/bin/:/usr/bin:/usr/sbin
```

Quadro 7 – grabmodules.sh

```
#!/bin/bash
# grabmodules: acessa servidores
# rsync e sincroniza os vários
# módulos configurados no arquivo
# grab.conf.
# Em seguida, compacta os
# arquivos e os armazena segundo
# um backup rotativo.
#
# Marcelo Barros
# (barros@smar.com.br)
# 20/Oct/2003
# inclui as definições globais
. /etc/grab/grab.defs
# pega a lista de servidores
# rsync e o tipo de sistema
# operacional
svr=(`cat $GRABDIR/grab.conf | grep -v "^#" | cut -d: -f1`)
so=(` cat $GRABDIR/grab.conf | grep -v "^#" | cut -d: -f3`)
# pega a lista de módulos
OLDIFS=$IFS
IFS=":"
rep=( $(cat $GRABDIR/grab.conf | grep -v "^#" | cut -d: -f2 | tr "\n" ":") )
IFS=$OLDIFS
# sincroniza os módulos de cada
# servidor de rsync
for i in $(seq 0 $((${#svr[@]}-1))); do
  echo "$GRABDIR/grabfiles.sh  ${svr[$i]}  ${rep[$i]} ${so[$i]}"
  $GRABDIR/grabfiles.sh ${svr[$i]} "${rep[$i]}" ${so[$i]}

done
# Executa o backup rotativo
. $GRABDIR/grabrotate.sh
```

processo do *rsync* será executado por um usuário com baixo nível de acesso (ex. *nobody* do Linux). Apesar de essa configuração ser mais segura, ela impede que o rsync defina o proprietário e o grupo originais dos arquivos no servidor de destino. Para que isso ocorra, é necessário incluir os seguintes parâmetros no módulo:

```
uid = root
gid = root
```

O *rsyncd.conf* permite definir outros parâmetros, como padrões de arquivos a ser excluídos da sincronização e informações de autenticação para acesso ao serviço. A documentação que acompanha o programa (digite *man rsyncd.conf*, no Linux) apresenta essas opções e muitas outras.

Grab: backup rotativo semanal

Ter e manter um bom sistema de backup é uma preocupação constante para qualquer administrador de redes. Aqui apresentamos uma solução para backup implementada em Shell Script que permite realizar o sincronismo com dois servidores de *rsync*, mas que pode ser facilmente estendido para mais máquinas, gerando uma única imagem com os dados provenientes de cada servidor. Essa imagem é compactada e armazenada de forma rotativa, de acordo com os dias da semana, gerando um sistema de backup automático e personalizável. Esse conjunto de scripts é chamado *Grab* e deve ser armazenado em */etc/grab*.

Os servidores de *rsync* são todos configurados para não aceitar escrita. Os arquivos de configuração são mostrados no Quadro 2 (máquina Linux), Quadro 3 (máquina Windows) e Quadro 4 (servidor de backup). Considere que, na máquina Windows, o *rsync* foi instalado no diretório *c:\rsync*.

Não se esqueça de deixar os arquivos */etc/grab/lin.pass* e */etc/grab/win.pass* com permissões de leitura e escrita somente para o dono do arquivo, isto é, *chmod 600 <arquivo>*. O *rsync* irá "reclamar" se essas permissões não estiverem corretas.

No arquivo */etc/crontab* da máquina 10.0.0.3 podem-se configurar a hora e periodicidade do backup. Adicione a seguinte linha ao crontab para sincronizar todos os servidores de *rsync* às 23:00 horas, de segunda a sexta, através do script *grabmodules.sh* (detalhado mais adiante):

```
00 23 1-5 * * * /etc/grab/➥
grabmodules.sh
```

O diretório em que os dados provenientes dos vários servidores serão armazenados pode ser configurado (variável *MIRRORDIR*). Para cada servidor de rsync existirá um subdiretório dentro de *MIRRORDIR* com o seu nome (ou IP), sendo que os módulos sincronizados são colocados dentro desse subdiretório em diretórios que levam o nome do módulo.

Quadro 8 – grabfiles.sh

```
#!/bin/bash
# grabfiles: sincroniza os módulos
# de um determinado servidor rsync
#
# Parâmetros:
#
# - endereço do servidor de rsync
# - lista de módulos, separados
# por espaços e entre aspas.
# Exemplo: "foo bar"
# - sistema operacional (windows
# or linux)
#
# Exemplo: ./grafiles.sh ➥
# foo.bar.com.br  "foo bar"  linux
# Exemplo: ./grabfiles.sh ➥
# 192.168.168.3   "projetos ➥
# visualc"  windows
#
# Marcelo Barros
# (barros@smar.com.br)
# 20/Oct/2003
#
# inclui as definições globais

. /etc/grab/grab.defs

# Pega os parâmetros da linha
# de comando
SERVER=$1
SO=$3
REPOSITORY=`echo $2 | tr -d "\""`

# Pega a data no formato ano/
# mes/dia. Ex: 20031020

TODAY=`date +%Y%m%e`

# Troca espaços por zero
TODAY=`echo $TODAY | tr " " "0"`

# Cria o diretório de mirror,
# caso não exista

if [ ! -d $MIRRORLOGDIR ]; then
  echo "mkdir -p $MIRRORLOGDIR"
  mkdir -p $MIRRORLOGDIR
fi

# Quantos módulos ?
m=`echo $2 | wc -w | tr -d " "` ➥
n=1

# Sincroniza cada módulo
for r in $REPOSITORY; do
  # Cria o diretório do módulo
  # caso não exista
  if [ ! -d $MIRRORDIR/$SERVER/$r ];
  then
    echo "mkdir -p $MIRRORDIR/➥
    $SERVER/$r"
    mkdir -p $MIRRORDIR/$SERVER/$r
  fi

  # windows ou linux ? as opções
  # podem mudar
  if [ "$SO" == "windows" ]; then
    OPTIONS=$RSYNCOPTWIN
  else
    OPTIONS=$RSYNCOPTLIN
  fi

  # Finalmente, roda o rsync
  echo "rsync $OPTIONS ➥
$SERVER::$r $MIRRORDIR/$SERVER/$r >& ➥
$MIRRORLOGDIR/$SERVER-$TODAY-$r.log"
  rsync $OPTIONS $SERVER::$r ➥
$MIRRORDIR/$SERVER/$r >& ➥
$MIRRORLOGDIR/$SERVER-$TODAY-$r.log
  echo "Confirmacao via email para ➥
$MAILTO ($SERVER-$TODAY-$r-$n/$m) ..."
  cat $MIRRORLOGDIR/$SERVER-
$TODAY-$r.log | mail -s ➥
"[MIRROR $n/$m] $SERVER-$TODAY-➥
$r.log" -c $MAILCP $MAILTO
  let n=$n+1
donez
```

Além disso, após um sincronismo completo, os dados são compactados (no caso, com o compactador *RAR*) e salvos dentro de um diretório com o nome do dia da semana correspondente (por exemplo, "Segunda", para sistemas em português). O nome dos arquivos compactados segue o padrão *<nome_do_servidor>-<nome_do_modulo>* e o local onde o backup se encontra é definido pela variável *BACKUPDIR*. Cada passo executado é registrado via email, permitindo uma verificação rápida do estado do procedimento de backup.

Os servidores de *rsync* que se deseja acessar, a lista de módulos em cada um deles e o tipo de sistema operacional são definidos no arquivo */etc/grab/grab.conf*, listado no Quadro 5.

A idéia ao dividir os servidores em Linux e Windows foi personalizar as opções que podem ser usadas com o *rsync*. Por exemplo, como o Windows não possui o mesmo sistema de permissões de acesso que o Linux, acontecem incompatibilidades se o mesmo conjunto de opções for usado.

O script */etc/grab/grab.defs* (Quadro 6) possui as definições globais para o sistema. É onde podem ser feitas as configurações de armazenamento.

O processamento é feito no script *grabmodules.sh*, que interpreta o arquivo *grab.conf* e realiza o sincronismo através do script *grabfiles.sh*. Finalmente, o armazenamento rotativo é executado pelo script *grabrotate.sh*, também chamado por *grabmodules.sh*. Estes scripts são mostrados nos Quadros 7 e 8, e devem ser gravados no diretório */etc/grab*.

Isso finaliza nosso exemplo. Mesmo sendo composto por vários scripts, o uso do Grab é simples e muitas melhorias podem ser adicionadas. Não se esqueça de deixar todos os scripts dentro do diretório */etc/grab* e de ajustar os arquivos de configuração de acordo com as suas necessidades. ■

Quadro 9 -- grabrotate.sh

```
#!/bin/bash
# grabrotate: Compacta a imagem
# e armazena no diretório
# correspondente ao dia
# da semana apropriado
#
# Marcelo Barros
# (barros@smar.com.br)
# 20/Oct/2003
# inclui as definições globais
. /etc/grab/grab.defs
# Pega o dia da semana
WEEKDAY=`date +%A`
# Cria o diretório de backup,
# caso não exista
if [ ! -d $BACKUPLOGDIR ]; then
  mkdir -p $BACKUPLOGDIR
fi
# Cria o diretório para o dia da
# semana, caso não exista
if [ ! -d $BACKUPDIR/$WEEKDAY ];
then
  mkdir -p $BACKUPDIR/$WEEKDAY
fi
# pegando a lista de servidores
# de rsync e os módulos
svr=(`cat $GRABDIR/grab.conf | ➥
grep -v "^#" | cut -d: -f1`)
OLDIFS=$IFS
IFS=":"
rep=( $(cat $GRABDIR/grab.conf | ➥
grep -v "^#" | cut -d: -f2 | tr ➥
"\n" ":" ) )
IFS=$OLDIFS
# Apaga o backup antigo. Muita
# calma nesta hora.
rm -fR $BACKUPDIR/$WEEKDAY/*

# Para cada servidor, compacta e
# armazena todos os módulos
for i in $(seq 0 $((${#svr[@]}-➥
1))); do

  # Separa a lista de módulos
  rep2=( $(echo ${rep[$i]} | tr ➥
-d "\"" ))

  # Só para indexar
  m=`echo ${rep[$i]} | tr -d ➥
"\"" | wc -w | tr -d " "`
  n=1
  for j in ${rep2[@]}; do

    # O RAR V2.8 tem um bug
    # relacionado a nomes de
    # arquivos com vários pontos
    # (usamos o IP como nome aqui).
    # Herança da versão Windows?
    # Trocamos '.' por '-'

    server=`echo ${svr[$i]} | tr ➥
"." "-"`
    echo "rar $RAROPTS ➥
$BACKUPDIR/$WEEKDAY/$server-$j ➥
$MIRRORDIR/${svr[$i]}/$j >& ➥
$BACKUPDIR/$WEEKDAY/$server-$j.log"
    rar $RAROPTS $BACKUPDIR/
$WEEKDAY/$server-$j ➥
$MIRRORDIR/${svr[$i]}/$j >& ➥
$BACKUPDIR/$WEEKDAY/$server-$j.log
    # Envia o log por email
    cat $BACKUPDIR/$WEEKDAY/
$server-$j.log | mail -s ➥
"[BACKUP $WEEKDAY $n/$m] $server-➥
$j.log" -c $MAILCP $MAILTO
    t
    let n=$n+1

  done
done
```


SOBRE OS AUTORES

Marcelo Barros de Almeida (barros@smar.com.br), 32 anos, é doutor em engenharia elétrica pela Universidade Federal de Minas Gerais (UFMG) e trabalha atualmente na Smar Equipamentos Industriais, desenvolvendo equipamentos embarcados para controle industrial baseados em tecnologia da Fieldbus Foundation.

Juliano Simões (juliano@axios.com.br), 36 anos, é mestre em Ciência da Computação pela Universidade de Birmingham (no Reino Unido) e trabalha atualmente como Gerente de Tecnologia do provedor de hospedagem web Central Server.

INFORMAÇÕES

[1] Site do Cygwin: *http://www.cygwin.com*

[2] Installing RSYNC no Windows 2000/NT: *http://www.tiarnan.phlegethon.org/rsyncntdoc.html*
Veja também: *http://optics.ph.unimelb.edu.au/help/rsync/rsync_pc1.html*

[3] Página oficial do projeto rsync: *http://rsync.samba.org/*

[4] Página de Andrew Tridgell. Contém uma cópia do algoritmo utilizado no rsync: *http://samba.org/~tridge/*

[5] Solução para Backup Multiplataforma Usando Software Livre. Juliano Simões. Apresentada durante a Comdex/2003: *http://simoes.pro.br/juliano/*

[6] FAQ oficial: *http://samba.org/rsync/FAQ.html*