

Leonardo José Zanotti

Tecnologia em Análise e Desenvolvimento de Sistemas

Administração de Sistemas – Professor Mauro Antônio Alves
Castro

Zabato – Zanotti's Backup Tool (Shell Script)

Curitiba
30/08/2020

SUMÁRIO

Introdução.....	03
Requerimentos.....	03
Instalação e execução.....	03
Backup Manual.....	04
Backup Automático.....	08
Agendando a execução.....	09
Conclusão.....	11

Zabato – Documentação

Introdução

Zabato é uma ferramenta de backup incremental, isto é, backup apenas de arquivos e pastas do diretório de origem que sofreram alguma modificação em relação aos arquivos que já estão no diretório de destino.

A ferramenta pode ser utilizada para realizar a sincronização de arquivos em intervalos regulares de tempo, a serem configurados pelo usuário. Desse modo, o principal uso da ferramenta consiste em realizar o backup de uma dada pasta do sistema em datas específicas de forma automática.

Este programa possui duas versões: automática (zabato-auto.sh) e manual (zabato-manual.sh). A versão automática será utilizada e analisada posteriormente na seção **Backup Automático**. Inicialmente será comentado sobre a execução da versão manual.

Requerimentos

Os requerimentos para o uso desta ferramenta é um sistema operacional UNIX ou um interpretador de shell script.

Instalação e Execução

A instalação do programa consiste em clonar o repositório no github e dar permissão de execução ao programa:

Clonando o repositório

```
$ git clone https://github.com/LeonardoZanotti/Zabato.git
```

Entrando na pasta do projeto

```
$ cd Zabato/
```

Dando permissão de execução

```
$ chmod +x zabato-manual.sh
```

Para executar o programa, basta usar:

Executando o script

```
$ ./zabato-manual.sh
```

Figura 1 – Tela inicial do programa.

```
mmmmmm      #          m          mm      /^( )^\
#           #          #mm      mm#mm      \,(..),/
m#  "  #  #mm      mm      #mm      V~~V
m"  m""#  #  #  m""#  #  #  #  #
##mmm  "mm"#  ##m#"  "mm"#  "mm  "#m#"

Zanotti backup tool

You want to make a backup from ./origem/ to ./destino/ ?  [y/n]
█
```

Backup Manual

Para o backup manual será utilizado o programa zabato-manual.sh.

Por padrão, o programa está considerando o diretório **./origem** como diretório a realizar o backup e **./destino** como diretório onde o backup será armazenado. Para se alterar esses diretórios deve-se editar as seguintes linhas do script:

Figura 2 – Diretórios do programa.

```
17  orig=./origem/      # origin dir
18  dest=./destino/     # destine dir
19  logfile=./logs.txt  # log file
```

A variável **orig** corresponde ao diretório de origem e **dest** corresponde ao diretório de destino. Na imagem também é possível ver a variável **logfile**, a qual corresponde ao arquivo de registros do backup.

Inicialmente o programa pede uma confirmação de diretórios, como pode se ver na figura 1. Ocorrendo a confirmação, ocorre um backup de teste utilizando **rsync**.

Figura 3 – Confirmação e backup de teste.

```
21  echo "You want to make a backup from ${orig} to ${dest} ?  [y/n]"
22  read resp
23  if [ $resp = 'y' ] || [ $resp = 'Y' ]; then
24      clear
25      echo 'Please confirm the following actions'
26      echo
27      rsync -auvn --progress --delete --exclude='.DS_Store' --log-file=$logfile $orig $dest  ## backup test to confirm the backup
28      echo
```

Na figura 3 o comando **rsync -auvn --progress --delete --exclude='.DS_Store' --log-file=\$logfile \$orig \$dest** é o responsável pelo backup teste, de modo que:

- **-a** → archive, realiza a cópia dos arquivos e mantém a assinatura dos mesmos.
- **-u** → update, só realiza a sincronização dos arquivos modificados.
- **-v** → verbose, aumenta a quantidade de informação que o usuário vê.
- **-n** → dry-run, realiza o backup sem fazer alterações (teste apenas).

- **--progress** → mostra o progresso durante a transferência.
- **--delete** → remove arquivos alheios do diretório de destino.
- **--exclude='.DS_Store'** → Não sincroniza este arquivo.
- **--log-file=\$logfile** → Define o arquivo de log como **logs.txt**.
- **\$orig** → Define o diretório de origem como **./origem/**.
- **\$dest** → Define o diretório de destino como **./destino/**.

Figura 4 – Backup de teste.

```
Please confirm the following actions

sending incremental file list
deleting file2.txt
./
file.txt
work.txt
img/
img/image.png

sent 208 bytes  received 45 bytes  506.00 bytes/sec
total size is 17  speedup is 0.07 (DRY RUN)

Transferring 12K ./origem/ to ./destino/
Continue? [y/n]
█
```

Este backup apenas simula como o processo ocorreria, desse modo o usuário pode verificar se o backup que será realizado está realmente correto. De acordo com a figura 4, vemos que o backup irá apagar **file2.txt**, por estar no diretório de destino mas não no de origem, e adicionar **file.txt**, **work.txt**, pasta **img** e imagem **image.png** que está dentro da pasta **img** ao diretório de destino.

Ocorrendo novamente a confirmação, o backup irá realmente realizar alterações nos diretórios:

Figura 5 – Backup do diretório de origem para destino.

```
Transferring 12K ./origem/ to ./destino/
Continue? [y/n]
y
sending incremental file list
deleting file2.txt
./
file.txt
work.txt
img/
img/image.png

sent 337 bytes  received 93 bytes  860.00 bytes/sec
total size is 17  speedup is 0.04
Backup done
```

Após o backup, vemos a alteração nos diretórios:

Figura 6 – Pastas antes do backup.

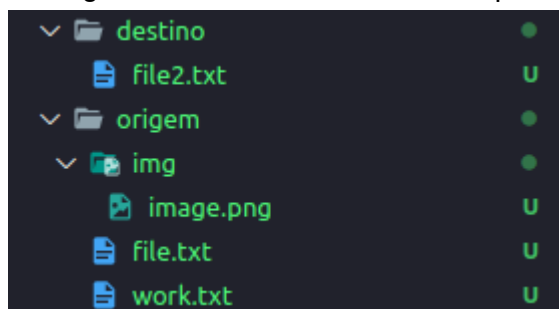
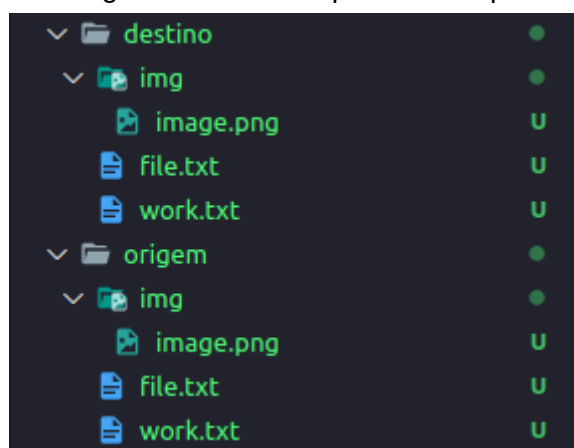


Figura 7 – Pastas após o backup.



O processo de backup é feito pelo mesmo comando do **rsync** já explicado, com a exceção de que é retirado o parâmetro **n**, para que as alterações sejam realmente feitas.

Figura 8 – Código de backup.

```
29 echo "Transferring" \ `du -sh $orig` \ `to` $dest
30 echo "Continue? [y/n]"
31 read resp
32 if [ $resp = 'y' ] || [ $resp = 'Y' ]; then
33     rsync -auv --progress --delete --exclude='.DS_Store' --log-file=$logfile $orig $dest    ## official backup
34 else
```

Caso em alguma das confirmações o usuário não digite **y** ou **Y** para confirmar, o backup será cancelado.

Figura 9 – Código para abortar o backup.

```

34     else
35         echo 'Aborting...'
36         exit
37     fi
38 else
39     echo 'Aborting...'
40     exit
41 fi
42
43 echo `date +%F %T`\ '- Manual backup done' >> $logfile
44 echo 'Backup done'

```

Figura 10 – Cancelamento de backup.

```

mmmmmm      #          m          mm      /^( )^\
              #          m          m      \,(..),/
              #          m          m      V~V
mmmmmm      #          m          mm      /^( )^\
              #          m          m      \,(..),/
              #          m          m      V~V
m# " # #mmmm mmm mmm mm#mm mmm
m" m""# # # m""# # # # #
##mmmm "mm"# ##m#" "mm#" "mm "#m#"

Zanotti backup tool

You want to make a backup from ./origem/ to ./destino/ ? [y/n]
n
Aborting...

```

Na Figura 9 também é possível perceber na linha 43 que temos, ao final do backup, a gravação de data e mensagem de sucesso no arquivo de logs para controle de backups.

No arquivo de logs ficam gravados todas as ações feitas durante o backup, arquivos apagados e criados, as pastas em que o programa entrou, etc.

Figura 11 – Arquivo de logs (logs.txt).

```

379 2020/09/03 12:05:02 [18907] building file list
380 2020/09/03 12:05:02 [18907] .d..t..... ./
381 2020/09/03 12:05:02 [18907] cd+++++++ img/
382 2020/09/03 12:05:02 [18907] sent 208 bytes received 45 bytes 506.00 bytes/sec
383 2020/09/03 12:05:02 [18907] total size is 17 speedup is 0.07 (DRY RUN)
384 2020/09/03 12:06:34 [19218] building file list
385 2020/09/03 12:06:34 [19218] *deleting file2.txt
386 2020/09/03 12:06:34 [19218] .d..t..... ./
387 2020/09/03 12:06:34 [19218] >f+++++++ file.txt
388 2020/09/03 12:06:34 [19218] >f+++++++ work.txt
389 2020/09/03 12:06:34 [19218] cd+++++++ img/
390 2020/09/03 12:06:34 [19218] >f+++++++ img/image.png
391 2020/09/03 12:06:34 [19218] sent 337 bytes received 93 bytes 860.00 bytes/sec
392 2020/09/03 12:06:34 [19218] total size is 17 speedup is 0.04
393 2020-09-03 12:06:34 - Manual backup done
394

```

Backup Automático

Para o backup automático, será usado o programa zabato-auto.sh.

Os requerimentos e processos de instalação e execução são os mesmos, com exceção de que onde lê-se “zabato-manual.sh” deve-se substituir por “zabato-auto.sh”.

Zabato-auto.sh é uma versão reduzida do zabato-manual.sh, pois seu propósito é ser usado para realizar os backups automaticamente por meio de um agendamento no sistema.

Figura 12 – Zabato-auto.sh.

```
4 orig=/home/leonardozanotti/origem/          # origin dir
5 dest=/home/leonardozanotti/Desktop/destino/    # destine dir
6 logfile=/home/leonardozanotti/Desktop/NovaPasta/logs.txt    # log file
7
8 rsync -auv --progress --delete --exclude='.DS_Store' --log-file=$logfile $orig $dest    ## official backup
9
10 echo `date +%F %T` \ '- Auto backup done' >> $logfile
```

Olhando para o código vemos apenas duas ações propriamente ditas (além da definição das variáveis orig, dest e logfile): na linha 8 temos o comando de backup usando rsync e na linha 10 temos a gravação do log de backup. Interessante notar que a declaração de variáveis no programa automático necessariamente tem que ser completo, sendo definido a partir do ‘/’ visto que é o sistema que o executará. No programa manual pode-se declarar as variáveis partindo do diretório do programa pois a execução é feita pelo usuário.

Como é o sistema quem executará esse programa, não há nada a ser exibido na tela e não é possível realizar a confirmação dos procedimentos (como no manual), desse modo o programa automático se resume a fazer o backup e gravar o log. Com isso é importante se atentar ainda mais a este código, visto que não há a confirmação de diretórios, o sistema apenas realizará o backup, e caso haja algum erro isso poderá gerar perda de arquivos.

Ao executarmos esse script para testar os diretórios podemos ver que está tudo correto, conforme indica a figura 13. Desse modo, podemos partir para o agendamento de execução do programa automático.

Figura 13 – Execução do programa zabato-auto.sh.

```
leonardozanotti@leonardozanotti:~/Desktop/Projetos-Linux/Administração de Sistemas/Projeto Final/Zabato$ ./zabato-auto.sh
sending incremental file list
created directory /home/leonardozanotti/Desktop/destino
./
file.txt
work.txt
img/
img/image.png
sent 324 bytes  received 148 bytes  944.00 bytes/sec
total size is 0  speedup is 0.00
```


Agendando a execução

Tendo o script `zabato-auto.sh` funcionando como desejado, basta agendar a sua execução. Para isso utiliza-se o Crontab para editar o arquivo que lista as tarefas a serem executadas pelo Cron.

Pequeno guia de Crontab:

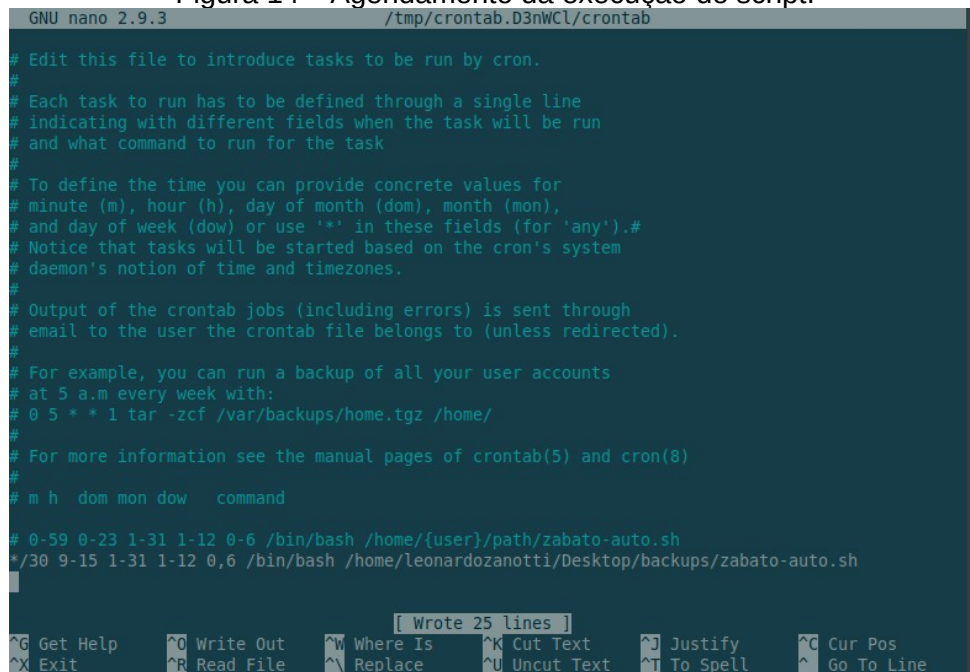
```
$ crontab -e → abre arquivo para agendar tarefa
$ crontab -l → lista tarefas agendadas
$ crontab -r → remove todas as tarefas
```

Agendando tarefas no Crontab:

```
minutos | horas | dias | meses | dias da semana | comando
0-59 0-23 1-31 1-12 0-6 echo "Olá" >> /home/{user}/text.txt
```

Desse modo, usamos o comando `$ crontab -e` para entrar no arquivo e editá-lo.

Figura 14 – Agendamento da execução do script.



```
GNU nano 2.9.3 /tmp/crontab.D3nWCL/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
# 0-59 0-23 1-31 1-12 0-6 /bin/bash /home/{user}/path/zabato-auto.sh
*/30 9-15 1-31 1-12 0,6 /bin/bash /home/leonardozanotti/Desktop/backups/zabato-auto.sh

[ Wrote 25 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

As linhas marcadas com `#` estão comentadas e não serão lidas pelo Cron. Com isso, vemos a última linha: `*/30 9-15 1-31 1-12 0,6 /bin/bash /home/leonardozanotti/Desktop/backups/zabato-auto.sh`. Essa linha diz para o Cron executar o comando `"/bin/bash /home/leonardozanotti/Desktop/backups/zabato-auto.sh"` (que é o mesmo que `./zabato-auto.sh` só que a nível de sistema) a cada 30 minutos, das 9h às 15h de todos os sábados e domingos de todos os meses. Obviamente que você leitor deve escolher os minutos do ano em que deseja o backup, basta alterar os campos dessa linha.

Pressiona-se `Ctrl + S` para salvar e `Ctrl + X` para sair e vemos a resposta do Crontab:

Figura 15 – Execução do script agendada.

```
leonardoazanotti@leonardoazanotti:~$ crontab -e
crontab: installing new crontab
leonardoazanotti@leonardoazanotti:~$
```

Com isso sabemos que conseguimos agendar a tarefa. Caso queiramos nos certificar do que escrevemos usamos `$ crontab -l`, o qual listará os agendamentos:

Figura 16 – Tarefas agendadas.

```
leonardoazanotti@leonardoazanotti:~$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
# 0-59 0-23 1-31 1-12 0-6 /bin/bash /home/{user}/path/zabato-auto.sh
*/30 9-15 1-31 1-12 0,6 /bin/bash /home/leonardoazanotti/Desktop/backups/zabato-auto.sh
leonardoazanotti@leonardoazanotti:~$
```

É interessante que, antes de se agendar uma tarefa nos dias e horários corretos, agendemos essa mesma tarefa a cada minuto apenas para nos certificar de que o Crontab realizará a execução de modo correto. Para isso, basta substituir cada campo por `*` e esperar a execução.

Figura 17 – Agendamento do script para teste.

```
GNU nano 2.9.3 /tmp/crontab.hy1yN0/crontab Modified
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
# 0-59 0-23 1-31 1-12 0-6 /bin/bash /home/{user}/path/zabato-auto.sh
* * * * * /bin/bash /home/leonardoazanotti/Desktop/backups/zabato-auto.sh
#
^G Get Help      ^O Write Out    ^W Where Is    ^K Cut Text     ^J Justify     ^C Cur Pos
^X Exit          ^R Read File    ^_ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

Assim que verificar que o backup foi executado corretamente, basta voltar os valores do campo aos corretos e está feito, nosso backup automático está corretamente agendado e agora nossos arquivos e pastas estarão em segurança.

Conclusão

A ferramenta de backup automático Zabato pode ser utilizada no dia a dia de qualquer usuário para garantir alguma segurança a arquivos importantes ao mesmo. Acredito que seja interessante que o processo de backup seja feito utilizando um dispositivo removível como destino, o qual costuma estar em `media/{user}/{dispositivo}`. Para isso basta apenas alterar a variável **dest** no programa.

Dependendo da necessidade, a ferramenta de backup manual pode ser mais indicada para o processo, em casos onde o backup é muito delicado, a ferramenta manual possui uma segurança maior com as confirmações e backup de teste, além de ser executada pelo usuário no momento apropriado.

De um modo geral, ambas as ferramentas cumprem o seu papel e estão de acordo com o esperado, podendo no futuro serem aprimoradas de modo a realizarem backups de várias origens para vários destinos por vez.