Capítulo 9 - Compactadores e agendamento de tarefas

9.1. Objetivos

Entender como empacotar e compactador arquivos/diretórios e como fazer agendamento de tarefas.

Função de cada um:

•tar - só empacota, não reduz tamanho, apenas junta todos os arquivos em um.

•gzip/bzip2 - compactador, ou seja, compacta um arquivo reduzindo seu tamanho.

No mundo GNU/Linux, é muito comum ver a extensão .tar.gz em arquivos de programas, e outros. Essa extensão é quando usamos o tar e o gzip juntos, pois aí, além de empacotar, também faremos a compactação!

O bzip2 faz uma compactação muito mais eficiente, ou seja, reduz mais que o gzip! Em compensação, ele demora um pouco mais para realizar essa compactação! Seria assim:

gzip - Rápido, porém não muito eficiente na redução de tamanho.

bzip2 - Lento, porém muito eficiente na redução de tamanho.

Então fazendo exemplo deles. Entre no diretório /tmp para mostrar como funciona:

```
# cd /tmp
```

Crie um arquivo relativamente grande para testar os dois compactadores. Utilize um shell script para criar um arquivo grande, por enquanto não se preocupe em entender perfeitamente o script. Script para criar um arquivo grande:

```
# while true
> do
> cat /etc/passwd >> /tmp/arquivo.txt
> done
```

Deixe esse script executando por uns 30 segundos! Onde os ">" não precisam ser digitados, pois vai aparecer (> é o prompt secundário).

Nesse script, existe um laço infinito deixando o arquivo.txt sempre com o conteúdo do /etc/passwd (que é um arquivo do sistema que guarda informações sobre cada usuário no sistema). Para parar basta dar CTRL+C para interromper a execução do script.

Então, depois de parar o loop (laço). Veja o tamanho do arquivo:

```
# cd /tmp
# du -h arquivo.txt
```

E aproveite e faz uma cópia do mesmo:

```
# cd /tmp
# cp arquivo.txt arquivo2.txt
```

Para testar os compactadores, vamos faça assim:

```
arquivo.txt - Para o gzip
arquivo2.txt - Para o bzip2
```

Lembrando que os dois precisam estar instalados na máquina.

```
# cd /tmp
# time gzip arquivo.txt
# time bzip2 arquivo2.txt
```

O comando time é apenas para marcar o tempo que o comando irá usar para terminar. Então, depois que o comando foi executado, e você poderá ver pelo time que o bzip2 demora bem mais que o gzip. Então, existirão dois arquivos no /tmp, um com a extensão gz e outro com bzip2:

```
#ls arquivo*
arquivo.txt.gz  arquivo2.txt.bz2
```

Agora, compare o tamanho dos dois arquivos:

```
#du -h arquivo*
68K     arquivo.txt.gz
24K     arquivo2.txt.bz2
```

Resumindo, veja que o gzip ganhou no tempo, mas perdeu em redução de tamanho. A ideia é ter gerado um arquivo grande, se está pequeno execute o script novamente e deixe mais tempo antes de dar CTRL+C.

O tar é um empacotador, para juntar vários arquivos ou diretórios em um só! Então vamos ao exemplo:

```
# cd /tmp
# tar -cvf bkp_etc.tar /etc
```

Onde:

-c - Para criar um backup

-v - (verbose) Para mostrar detalhes para você na hora de criar

-f - Para indicar o nome do arquivo. Essa opção sempre vem por último, pois é ela quem define o nome do arquivo.

Então pode fazer o teste de tamanho e ver que não reduziu tamanho:

```
# du -h /etc
13M     /etc

# du -h bkp_etc.tar
13M     /etc
```

Para dizer para o tar que ele deve executar juntamente com um compactador é necessário acrescentar uma opção ao comando:

z- Para compactar com GZIP

j - Para compactar com BZIP2

Então o comando mudaria para:

Tar com Gzip:

```
# cd /tmp
# tar -cvzf bkp_etc.tar.gz /etc
```

Onde:

c - Para criar um backup

v - Para mostrar detalhes para você na hora de criar

z - Para compactar com o GZIP

f- Para indicar o nome do arquivo de backup

Tar com Bzip2:

```
# cd /tmp
# tar -cvjf bkp_etc.tar.bz2 /etc
```

Onde:

c - Para criar um backup

v - Para mostrar detalhes para você na hora de criar

j - Para compactar com o BZIP2

f - para indicar o nome do arquivo de backup

Pode ver que só muda o j e o z. O resto são as mesmas opções. Com esses 3 comandos que eu passei, existirão 3 arquivos dentro do meu /tmp:

```
# ls /tmp | grep bkp
 bkp_etc.tar
 bkp_etc.tar.gz
 bkp_etc.tar.bz2
```

Compare seus tamanhos:

```
# du -h bkp*
9.3M    bkp_etc.tar
2.1M    bkp_etc.tar.bz2
2.5M    bkp_etc.tar.gz
```

Como são arquivos de BACKUP, é preciso ter certeza que esses arquivos estão realmente bons. Então, teste a integridade desses arquivos, e, para isso use a opção t:

```
# tar -tvf bkp_etc.tar
 # tar -tvzf bkp_etc.tar.gz
 # tar -tvjf bkp_etc.tar.bz2
```

No qual t é a opção que testa. Depois, para descompactar esses arquivos só mude para a opção para x que é de extract:

```
# tar -xvf bkp_etc.tar
# tar -xvzf bkp_etc.tar.gz
# tar -xvjf bkp_etc.tar.bz2
```

Caso queira descompactar no / para substituir o /etc antigo no caso de um backup especifique isso com o -C:

```
# tar -xvf bkp_etc.tar -C /
 # tar -xvzf bkp_etc.tar.gz -C /
 # tar -xvjf bkp_etc.tar.bz2 -C /
```

9.2. Agendadores de tarefas do GNU/Linux

A crontab é utilizada para agendar comandos para serem executados periodicamente, ao contrário do comando at que executa comandos pontualmente. Há dois tipos de crontab: de usuários e sistema.

Ambas são arquivos que contém tabelas com informação de quando o comando especificado deve ser executado, sendo que cada linha corresponde a um único agendamento. A crontab é gerenciada pelo daemon crond que cada minuto verifica se há algum agendamento que deve ser executado e executa. Antes de partir para o Cron, veremos como usar o at.

9.2.1. Agendador at

Agende para 10 minutos no futuro a listagem do diretório /etc/ redirecionando a saída para o terminal 11 (modo texto) e em seguida imprima no mesmo terminal a data e a hora.

Por exemplo: vamos supor que estou em outubro (10), dia 28, ano 2008, e a hora é 15:00 e quero que a tarefa seja executada às 20:00, então faço:

```
# at 20:00 10/28/2008
at > ls –color /etc/ > /dev/tty11
at > echo $(date +”%H:%m %M/%d/%Y”) >> /dev/tty11
at > ^d
^d = CTRL + D
```

Agendada essa tarefa, confirme-a listando todos os agendamentos pendentes:

```
# atq
```

Explore o diretório onde ficam os agendamentos:

```
# cd /var/spool/cron/atjobs
# ls -la
```

Realize outro agendamento, para executar em 15 minutos, para que possa aprender como apagá-lo:

```
# at HH:mm MM/DD/YYYY
 at > echo “Teste” > /tmp/at.out
 at > ^d
```

Liste os agendamentos correntes e verifique que um novo arquivo foi criado no diretório de spool do at:

```
# atq
 4       Tue Feb  5 14:30:00 2008 a root
```

Remova o agendamento:

```
# atrm 4
```

9.2.2. Crontab

Todos os usuários podem agendar tarefas no sistema. Mas ele terá que ter permissão para executar o que está agendando.

Para editar o crontab do usuário padrão:

```
# crontab -e
```

(Para todos os usuários que queiram usar o cron)

Onde -e significa edit. Isso fará com que se abra um arquivo em branco, caso não tenha nenhuma tarefa agendada ainda, para que você possa adicionar uma tarefa. Voltando no arquivo que abriu do comando crontab -e. Nesse arquivo em branco, temos que adicionar uma nova tarefa seguindo essa estrutura:

```
*    *    *    *    *    comando/tarefa
```

Ao entrar no arquivo e necessário editar 6 colunas que podem estar separadas por TAB ou espaço:

Coluna Função
1 Minuto
2 Hora
3 Dia do mês
4 Mês
5 Dia da semana
6 Programa que será executado

Onde:

Coluna Função
Minuto -> 0-59
Hora -> 0-23
Dia do mês-> 1-31
Mês -> 1-12
Dia da semana -> 0-7 (o "0" e "7" é domingo, "1" segunda)

Exemplo:

```
#crontab -e
00    23    *    *    *    halt
```

O comando acima significa desligar a máquina, todos os dias (3º Campo com *) , de todos os meses (4º Campo com *) sendo qualquer dia da semana (5º Campo com *) as 23h00.

É importante falar que a regra sempre será um AND em cada campo, ou seja, a tarefa só será executada se atender a todos os campos da regra.

Por exemplo:

```
00    12    25    12    0    echo "Hoje é Domingo" >> /dev/tty2
```

A tarefa acima é para mandar uma mensagem para o terminal 2 no modo texto, e ela só será executa se for 12h00 do dia 25 de dezembro. E esse dia precisará ser domingo, caso contrário a mesma não é executada.

Para ver as tarefas que foram agendas pelos usuários (root) digite:

```
# crontab -l -u root
00     22     *     *     *     halt
```

Onde:

-l Para listar

-u Para especificar o usuário

Quando mandar salvar a regra, ou a tarefa agendada, a mesma irá para um arquivo com o nome do meu usuário:

```
# ls /var/spool/cron/crontabs/root
```

Então basta dar um cat nesse arquivo para ver todas que você agendou lá.

Atenção! Não apague ou edite o seu agendamento dentro desse diretório, use os comandos para fazer isso.

Agora, veja como agendar para que esse script funcione no CRON, mas antes uma rápida revisão:

```
#crontab -e
```

Veja a linha para agendar o backup.

```
00 21 * * 1-5 /root/backup.sh
```

Veja que, no cron, é preciso colocar o caminho completo de onde está o script: /root/backup.sh

Suponha que você queira que um backup seja feito de 2 em 2 minutos:

```
*/2 * * * * /root/backup.sh
```

Entendendo os operadores do formato de agendamento:

vírgula (,) especifica uma lista de valores, por exemplo: "1,3,4,7,8";
– especifica um intervalo de valores, por exemplo: 1-15 (de 1 a 15);
** especifica todos os valores possíveis;*
*/ especifica pulos de valores, por exemplo, se no campo hora utilizarmos */3 o comando será executado às 0,3,6,9,12,15,18,21.*

Os arquivos de crontab dos usuários ficam em:

```
# cd /var/spool/cron/crontabs
```

Já a crontab do sistema é encontrada no /etc/crontab e já possui agendamentos para realizar as tarefas que se encontram nos diretórios /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly e /etc/cron.monthly.

A sintaxe de agendamentos de sistema é quase idêntico ao do usuário, à exceção que a crontab do sistema possui um campo USUARIO a mais:

```
minuto hora dia mês dia da semana USUARIO comando
```

Repare também que dentro do arquivo /etc/crontab existem 4 agendamentos já definidos: cron.hourly, cron.daily, cron.weekly e cron.monthly. Para que o script seja executado diariamente:

```
# cp /root/backup.sh /etc/cron.daily/backups
```

Para que o script seja executado semanalmente:

```
# cp /root/backup.sh /etc/cron.weekly/backups
```

Depois de adicionar o script dentro dos diretórios será necessário reiniciar o daemon do cron (Debian):

```
# /etc/init.d/cron stop
# /etc/init.d/cron start
```

Leitura sugerida:

```
# man 5 crontab
```