BaTBU: tenere i vostri dati al sicuro.

Bertera Pietro, Giangualano Matias, Branca Luca v0.1 settembre 2004

Questo documento spiega come e' possibile mantere delle copie di dati importanti ottimizzando le risorse di calcolo e di spazio. Per approfondire l'argomento verra' spiegato e commentato il funzionamento di BaTBU's Not Trivial Backup Utility

Contents

1 Il succo 2

1.1 Perchè? . . . 2

2 Usare rsync per creare i backup 3

2.1 Copiare con rsync . . . 3

2.2 Usare –delete . . . 3

2.3 Siete pigri? usate cron . . . 3

3 Backup incrementali con rsync 3

3.1 1000 file nello spazio di 1: gli hard links . . . 4

3.1.1 E se uso **rm** ? . . . 4

3.2 Risparmiare spazio: cp -al . . . 4

3.3 Il minestrone è pronto . . . 4

4 Isolare il backup 5

4.1 Partizione/disco/macchina . . . 5

4.1.1 In una partizione separata . . . 5

4.1.2 In un disco separato . . . 5

4.1.3 In una macchina separata . . . 5

5 Nessuno scriva! 5

5.1 Montare in rw solo quando serve . . . 6

5.2 Creare un wrapper con una condivisione. . . . 6

6 BaTBU's Not Trivial Backup Utility 6

6.1 Specifiche . . . 6

6.2 Sintassi . . . 7

6.2.1 Esempi . . . 7

7 Esempio di Utilizzo in ambienti di produzione 8
7.1 Backup di due alberi di directory automatizzato: 8
7.1.1 Topologia: 8
7.1.2 Spazio: 9
7.1.3 Tempo: 9
7.1.4 Risultati: 9

8 Codice: BatBU 11
8.1 Appendice A: autenicazione SSH tramite chiave RSA 19

1 Il succo

Questo documento descrive un metodo per gestire degli snapshot-style backup a rotazione automatici in un sistema Unix, con esempi specifici su un sistema GNU/Linux.

Gli snapshot backup sono una feature di molti file server industriali di alto livello: creano backup completi multipli durante la giornata senza overhead di spazio o di potenza di calcolo.

Tutti gli snapshot **devono** essere salvati su un file system in sola lettura e devono essere accessibili agli utenti tramite delle directory speciali.

Ottimizzando le techinche e gli strumenti utilizzati durante il processo di backup e' possibile avere un numero a piacere di copie dei dati occupando poco piu' dello spazio occupato da una copia dei dati. Tutto questo utilizzando solamente delle utility stanard e il programma rsync.

Se configurato correttamente questo metodo puo' proteggere da guasti hardware, da compromissioni del file system, da accessi indesiderati, pio' anche e ettuare il backup di una intera rete di macchine in modo automatico.

1.1 Perchè?

Immaginate

- Se sovrascrivete o cancellate un file importante su cui dovete lavorare
- Se perdete tutti i dati di un disco a causa di un guasto
- Se il vostro web server viene violato da un cracker
- Se esportate delle directory Samba e il virus sulla macchina del vostro collega giocherella con qualche bit dei vostri file

Non mi sarebbe utile una directory **/backups** in cui trovate le copie complete dei vostri file salvati ad intervalli di un ora?

Non vi sarebbero utili delle copie dei vostri dati di ieri, l'altro ieri e due giorni fa ? o magari delle copie di settimana scorsa e due settimane fa?

2 Usare rsync per creare i backup

rsync e' un software GPL scritto da Andrew Tridgell e Paul Mackerras. E' possibile prelevarne i sorgenti all' URL http://rsync.samba.org. La peculiarita' di rsync è l'e cenza nella sinconizzazione di alberi di file su reti ma lavora benissimo anche in locale.

2.1 Copiare con rsync

Sopponiamo di avere una direcotory chiamata **sorgente** e di volerne creare il backup in **destinazione**. Per realizzare questo:

```
rsync -c sorgente/ destination/
```

Il comando e' equivalente a:

```
cp -a sorgente/. destinazione/
```

l'unica di erenza è che rsync e' molto piu' ottimizzato: copia due file solamente se sono diversi.

Una cosa interessante e' la possibilita' di eseguire la stessa cosa tra due macchine remote sopra ssh:

```
rsync -a -e ssh sorgente/ user@host:/destinazione
```

2.2 Usare –delete

Se un file si trova sia in **sorgente/** che in **destinazione/** e il file viene eliminato da **sorgente/** probabilmente e' utile che venga eliminato anche in **destinazione/**.

```
rsync -a --delete sorgente/ destinazione/
```

2.3 Siete pigri? usate cron

Il primo ostacolo in una buona strategia di backup e' la pigrizia umana. Fortunatamente si puo' rimediare a questo facendo lavorare **cron** per noi.

Ad esempio se si desidera e ettuare il backup alle 2 e 40 AM basta editare la crontable nel seguente modo:

```
40 2 * * * root rsync -a --delete source/ destination/
```

3 Backup incrementali con rsync

Ovviamente creare una intera copia di un grosso file system ha bisogno di un processo oneroso sia di tempo che di risorse computazionali e di banda se si lavora con 2 macchine separate.

E' buona pratica creare un backup intero una volta alla settimana o una volta al mese e poi salvare solo le di erenze negli altri giorni. Questa tecnica prende il nome di backup incrementali.

3.1 1000 file nello spazio di 1: gli hard links

Quando si pensa al nome di un file si pensa che sia il file stesso ma in realta' il nome e' un **hard link** overo un riferimento all'inode relativo a quel file. Lo stesso file puo' avere uno o piu' hard link verso se stesso. Ad esempio una directory ha almeno 2 hard link: il nome della directory e il **.**. Anche i file **..** contenuti in una eventuale sottodirectory sono degli hard link alla directory madre. E' possibile vedere quantio hard link ha un file tramite il comando **stat**.

Si possono creare hard link con il comando **ln**. Si puo' verificare che due file sono in realta' degli hard link perche' puntano allo stesso **inode**.

```
scrofa:~# touch pippo
scrofa:~# ln pippo pluto
scrofa:~# ls -i pippo
130341 pippo
scrofa:~# ls -i pluto
130341 pluto
scrofa:~#
```

Fare **ln pippo pluto** ha lo stesso risultato di **cp pippo pluto** ma con delle fondamentali di erenze:

- Il contenuto del file e' immagazzinato una sola volta, non si utilizza il doppio dello spazio
- Se si modifica **pippo** la modifica avviene anche su **pluto** e vice versa
- Questo vale anche se si modificano i permessi
- Se si sovrascrive **pippo** copiandoci un terzo file sopra si sovrascrve anche **pluto**. Per evitare questo si puo' dire a **cp** di fare l'**unlink** prima di sovrascriverlo con l'opzione **--remove-destination**.

3.1.1 E se uso **rm** ?

rm non rimuove realmente il contenuto del file ma solamente l'hard link al contenuto, quindi un file viene rimosso quando il suo numero di link e' 0.

3.2 Risparmiare spazio: cp -al

E' stato detto prima che creare un hard link e' simle al processo copia, nelle standard GNU coreutils il comando cp ha l'opzione **-l** che crea un hard link anziche' una copia.

Un'altra opzione interessante e' **-a** (archivio) che copia ricorsivamente e mantiene i permessi, il proprietario e la data dei file.

Quindi la combinazione **cp -al** sembra che faccia una copia di un intero albero di directory ma in realta crea solo degli hard link e occupa molto meno spazio. Se si restringe l'operazione di copia in aggiungere e rimuovere (unlink) solo i file che cambiano sia ha l'illusione di avere un backup completo. Per l'utente finale l'unica di erenza e' che occupa molto meno spazio e ci vuole molto meno tempo a generarlo.

3.3 Il minestrone è pronto

E' possibile combinare **rsync** e **cp -al** per creare completi backup multipli in un filesystem senza avere bisogno di dischi multipli e di gradi quantita' di spazio.

```
rm -rf backup.3
mv backup.2 backup.3
mv backup.1 backup.2
cp -al backup.0 backup.1
rsync -a --delete sorgente/  backup.0/
```

Se i comandi sopra vengono eseguiti giornalmente allora **backup.0, backup.1, backup.2,** e **backup.3** sono rispettivamente la copia completa di **sorgente** di oggi, ieri, l'altro ieri e di 2 giorni fa.

Il vantaggio e' che lo spazio occupato non e' 4 volte **sorgente** ma una volta **sorgente** piu' lo spazio totale delle modifiche negli ultimi tre giorni. Inoltre da **sorgente/** a **destinazione/** vengono copiati solo i file che hanno subito modifiche.

4 Isolare il backup

Se si pensa di mantenre i backup in una directory dello stesso filesystem questo il modo migliore per perdere sia i dati che i backup.

4.1 Partizione/disco/macchina

4.1.1 In una partizione separata

Se il backup e' in una partizione separata sara' al riparo da danni subiti al filesystem principale; se lo spazio della partizione di backup viene riepito totalmente il sistema non cessera' di funzionare. E' importante mantenere i backup in una partizioe separata e montata in read only per evitare cancellazioni o sovrascritture involontarie.

4.1.2 In un disco separato

Se i backup stanno in un disco separato si e' al riparo anche da guasti hardware sul dispositivo: se un disco vi prende fuoco tutte le partizioni di esso verranno compromesse. Spesso puo' essere utile valutare di configurare un sitema RAID.

RAID e' ben supportato da Linux e il metodo qui descritto permette di creare anche degli snapshot su un sistema RAID.

4.1.3 In una macchina separata

Se si possiede una macchina inutilizzata anche di basse prestazioni puo' essere trasformata tranquillamente in backup server dedicato. Rendendola standaolne, mettendola in un posto separato, in una stanza o magari un edificio separato.

La macchina sorgente puo' esportare le directory che si vogliono copiare via NFS o Samba su di un interfaccia dedicata collegata al backup server oppure molto meglio sulla macchina sorgente viene configurato un server ssh.

5 Nessuno scriva!

Per essere utile un backup deve essere accessibile ababstanza facilmente.Ovviamente non e' una buona idea avere la directory di backup in un posto pubblico e accessibile in read-write. Putroppo il processo di backup

implica una scrittura, quindi richiede i permessi di write.

In un mondo ideale si potrebbe montare il backup in sola lettura in un mount-point pubblico e allo stesso tempo avere montato il backup in rw in un mount-point privato.

5.1 Montare in rw solo quando serve

Una soluzione puo' essere quella di montare il backup in rw prima di eseguire il backup e al termine dell'operazione rimontarlo in ro. Questa soluzione introduce una vulnerabilità: se durante la fase di backup qualche processo apre un file del backup in scrittura impedisce al processo di backup di rimontare in ro il filesystem.

Si puo' ovviare a questo in due modi:

- Utilizzare **fuser** per uccidere i processi che stanno usando il dispositivo di backup
- Montare il dispositivo durante la fase di backup in un luogo privato non accessibile agli utenti

A seconda della frequenza delle esecuzioni dei backup e degli accessi degli utenti su puo' valutare di scegliere una o l'altra opzione

5.2 Creare un wrapper con una condivisione.

Un'altra soluzione un po' macchinosa potrebbe essere quella di montare in rw il dispositivo e poi condividerlo con Samba o NFS impedendo la scrittura.

6 BaTBU's Not Trivial Backup Utility

6.1 Specifiche

BaTBU's Not Trivial Backup Utility è uno script per la gestione di backup completi, incrementali a rotazione periodica. Permette anche di sincronizzare delle directory remote in modo sicuro.

Di seguito una lista delle features dello script

- L'interfaccia con l'utente è la linea di comando per permettere l'inclusione dello script all'interno di altri script o di file di configurazione per l'automazione dell'esecuzione (ad es. cron).
- I backup devono essere salvati su di un dispositivo di erente dalla sorgente per evitare che compromissioni del filesystem della sorgente possano danneggiare i backup (è fortemente consigliato utilizzare un disco di erente per salvaguardarsi da eventuali hardware failure).
- Il dispositivo su cui vengono memorizzati i backup viene montato in sola lettura (tranne durante le operazioni di backup).
- Il trasferimento dei dati avviene in modo sicuro tramite il protocollo Open SSH e l'autenticazione con chiavi RSA (o DSA).
- La copia dei dati avviene in modo incrementale trasferendo solo i file modificati rispetto al backup precedente.
- L'occupazione di spazio su disco Ã¨ ottimizzata creando degli hard link ai file.
- Gestisce la rotazione dei backup in modo automatico creando dei nomi sequenziali.

- Tutta l'attivita' di backup viene registrata nei log di sistema e tramite stampe sullo stdout.
- Permette di escludere determinati file dall'operazione di backup.
- Permette di configurare la directory di destinazione ed il nome di ogni backup.

6.2 Sintassi

```
backup device mount_point source_dir [-d destination_dir] [-N backup_name] [-m max_backups] [-f eclu
```

BaTBU prevede tre parametri obbligatori:

- **device**: dispositivo a blocchi su cui viene archiviato il backup
- **mount_point**: mount point in cui viene montato il dispositivo **device**
- **source_dir**: radice dell'albero di directory di cui occorre fare il backup, puo' essere una directory locale o remota, in questo caso si veda l'opzione **-s**

I parametri opzionali sono:

- **-d destination_dir**: specifica la directory di destinazione (sotto **mount_point**)
- **-N backup_name**: il nome da assegnare al backup (una sottodirectry di **destination_dir** se esite)
- **-m max_backups**: il numero massimo di backup
- **-f exclude_file**: il file contenente la lista dei file da escludere durante la sincronizzazione
- **-s utente@host**: utente e host per l'accesso alla macchina remota tramite ssh (**utente** deve avere i permessi di lettura di **source_dir** su **host**)
- **-w**: mantiene il device montato in read/write (il device deve giÃ essere montato in rw)
- **-v**: rende piu' prolisso l'output dello script

6.2.1 Esempi

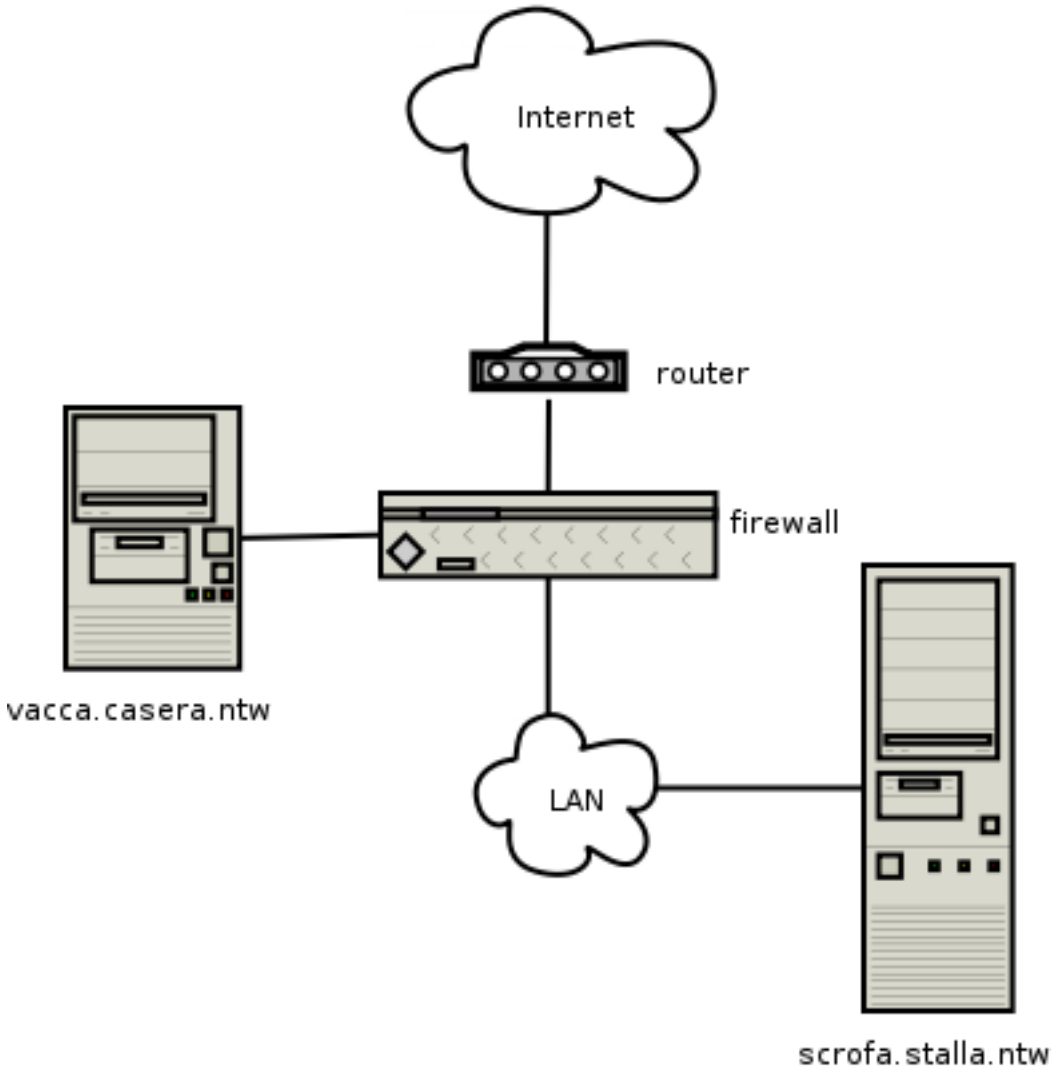
Seguono alcuni esempi di utilizzo:

Backup in locale: Backup di "/home/" sul disco hdd1 nella directory "backup-home" con nome "homebk", montare hdd1 in "/mnt/hdd1"

```
# batbu /dev/hdd1 /mnt/hdd1 /home -d backup-home -N homebk
```

Backup di "/home/" sul disco hdd1 nella directory "backup-home" con nome pari alla data odierna nel formato "yyyy-mm-gg", montare hdd1 in "/mnt/hdd1"

```
# batbu /dev/hdd1 /mnt/hdd1 /home -d backup-home -N ‘date +"%Y-%m-%d"‘
```



Backup in remoto Backup della directory ”/var/lib/mysql” della macchina ”vacca.casera.ntw” sul disco hdd1 nella directory ”vacca-database”, sulla macchina remota occorre fare il login come utente ”pastore”. hdd1 deve essere montato in /home/disk

```
# batbu /dev/hdd1 /home/disk /var/lib/mysql -s pastore@vacca.casera.ntw -d vacca-database
```

7 Esempio di Utilizzo in ambienti di produzione

7.1 Backup di due alberi di directory automatizzato:

7.1.1 Topologia:

Occorre mantere gli snapshot delle directory **/var/www/** e **/var/lib/mysql/** su **vacca**. Il backup server è **scrofa.stalla.ntw** il dispositivo su cui e ettuare i backup e' **/dev/hdd1** e il suo mount-point e' **/backups1**. I due backup verranno chiamati rispettivamente **vacca-www** e **vacca-mysql**.

Le due macchine sono separate da un firewall. Su **vacca** è installato e configurato per l'autenticazione tramite certificato RSA il server Open SSH.

7.1.2 Spazio:

/var/www/

```
vacca:~# du -sh /var/www/
405M    /var/www/
```

/var/lib/mysql

```
vacca:~# du -sh /var/lib/mysql/
33M     /var/lib/mysql/
```

Quindi **/var/www/** occupa 405 MB e **/var/lib/mysql** occupa 33MB.

7.1.3 Tempo:

Per quanto riguarda **/var/www/** si pensa di eseguire i backup orari dalle 8:30 alle 20:30 ad intervalli di tre ore. Il backup giornaliero avviene alle 12:30.

La directory **/bar/lib/mysql** viene salvata sempre ogni tre ore ma partendo dalle 8:00 fino alle 20:00. Il daily-backup avviene alle 12:00.

La rotazione dei backup orari avviene su di un numero massimo di 10 backup. Mentre i massimi backup giornalieri sono 5.

Per eseguire automaticamente i backup occorre aggiungere le seguenti linee ala cron-table (**/etc/crontab**):

```
30 8-20/3       * * *   root    batbu /dev/hdd1 /backups1 /var/www -d vacca-www -N hourly -m 10 -s r
0 8-20/3 * * *  root    batbu /dev/hdd1 /backups1 /var/lib/mysql -d vacca-mysql -N hourly -m 10 -s r
30 12   * * *   root    batbu /dev/hdd1 /backups1 /var/www -d vacca-www -N daily -m 5 -s root@192.16
0 12    * * *   root    batbu /dev/hdd1 /backups1 /var/lib/mysql -d vacca-mysql -N daily -m 5 -s roo
```

Per rendere e ettive le modifiche occorre fare ripartire il demone **cron**:

```
scrofa:/# /etc/init.d/cron restart
Restarting periodic command scheduler: cron.
```

7.1.4 Risultati:

Dopo due giorni di uptime di **scrofa**:

```
scrofa:/# ls -lh /backups1/vacca-mysql/
total 52K
drwx------  35 mysql mysql 4.0K Sep 18 12:00 daily.0
drwx------  35 mysql mysql 4.0K Sep 17 12:00 daily.1
drwx------  35 mysql mysql 4.0K Sep 19 11:00 hourly.0
drwx------  35 mysql mysql 4.0K Sep 19 08:00 hourly.1
drwx------  35 mysql mysql 4.0K Sep 18 20:00 hourly.2
drwx------  35 mysql mysql 4.0K Sep 18 17:00 hourly.3
drwx------  35 mysql mysql 4.0K Sep 18 11:00 hourly.4
```

```
drwx------  35 mysql mysql 4.0K Sep 18 08:00 hourly.5
drwx------  35 mysql mysql 4.0K Sep 17 20:00 hourly.6
drwx------  35 mysql mysql 4.0K Sep 17 17:00 hourly.7
drwx------  35 mysql mysql 4.0K Sep 17 11:00 hourly.8
drwx------  35 mysql mysql 4.0K Sep 17 08:00 hourly.9
```

Lo spazio occupato da un solo backup:

```
scrofa:~# du -sh /backups1/vacca-mysql/hourly.2
33M     /backups1/vacca-mysql/hourly.2
```

Mentre lo spazio occupato da 13 backup di **/var/lib/mysql**

```
scrofa:/# du -sh /backups1/vacca-mysql/
68M    /backups1/vacca-mysql/
```

Giustamente e' il doppio perche' ci sono i backup orari e i giornalieri.

Per quando riguarda **/var/www/**

```
scrofa:~# ls -lh /backups1/vacca-www/
total 48K
drwxr-xr-x  40 www-data www-data 4.0K Sep 18 12:49 daily.0
drwxr-xr-x  40 www-data www-data 4.0K Sep 17 12:41 daily.1
drwxr-xr-x  40 www-data www-data 4.0K Sep 19 11:41 hourly.0
drwxr-xr-x  40 www-data www-data 4.0K Sep 19 08:41 hourly.1
drwxr-xr-x  40 www-data www-data 4.0K Sep 18 20:48 hourly.2
drwxr-xr-x  40 www-data www-data 4.0K Sep 18 17:44 hourly.3
drwxr-xr-x  40 www-data www-data 4.0K Sep 18 11:43 hourly.4
drwxr-xr-x  40 www-data www-data 4.0K Sep 18 08:41 hourly.5
drwxr-xr-x  40 www-data www-data 4.0K Sep 17 20:41 hourly.6
drwxr-xr-x  40 www-data www-data 4.0K Sep 17 17:42 hourly.7
drwxr-xr-x  40 www-data www-data 4.0K Sep 17 11:43 hourly.8
drwxr-xr-x  40 www-data www-data 4.0K Sep 17 08:42 hourly.9
```

Un singolo backup di **/var/www/** occupa:

```
scrofa:~# du -sh /backups1/vacca-www/hourly.4
405M    /backups1/vacca-www/hourly.4
```

I dodici backup occupano:

```
scrofa:~# du -sh /backups1/vacca-www/
1.7G    /backups1/vacca-www/
```

In **/var/www/** l'overhead introdotto dallo spazio occupato da un singolo hard link seppur poco si fa sentire a causa del gran numero di file:

```
scrofa:~# find /backups1/vacca-www/hourly.8 -type f |wc -l
44141
```

Mentre in **/var/lib/mysql/**:

```
scrofa:~# find /backups1/vacca-mysql/hourly.2 -type f |wc -l
4818
```

La directory **/backups1** e' accessibile agli utenti tramite rete con una condivisione Samba in read-only /etc/samba/smb.conf

```
...
[backups]
        comment = Backups
        path = /backups1
        read only = yes
        valid users = @rolando,@pietro
...
```

8 Codice: BatBU

```
#!/bin/bash
################################################################################
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU Library General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
################################################################################

#
# log : stampa errori, warning e messaggi.
# parametri:    - $1 messaggio
#               - $2 codice d'errore:   se -1 stampa come prefisso "ERROR:" e termina lo script con
#                                       se 0 stampa il messaggio senza alcun prefisso
#                                       se 1 stampa il messaggio con il prefisso "WARNING:"
#
#       la stampa dei messaggi avviene sia a video che nei log di sistema.
#       Nella stampa nei log di sistema il messaggio viene anteceduto dal prefisso "BaTBU" per facil
#
#       Es.: ricerca in /var/log/messages di errori generati dallo script
#
#               eta:/home/pietro/backup# cat /var/log/messages |grep BaTBU
#               Sep 17 09:37:57 eta pietro: BaTBU::  Starting backup script...
```

```
#               Sep 17 09:37:57 eta pietro: BaTBU:: ERROR: no args
#

function log()
{
        PREFIX="BaTBU"
        if [ $2 ]
        then
                case $2 in
                        -1)
                                MESSAGE="ERROR:"
                                ;;
                        0)
                                MESSAGE=""
                                ;;
                        1)
                                MESSAGE="WARNING:"
                                ;;
                esac
                logger "$PREFIX $MESSAGE $1"
                echo "$MESSAGE $1"
                if [ $2 = -1 ]
                then
                        logger "$PREFIX Exit with status $2"
                        echo "Exit."
                        exit
                fi
        else
                logger "$PREFIX $1"
        fi
}

#start
#stampa messaggio di start
log "Starting backup script..." 0

#controlla se ci sono parametri tramite la variabile speciale $#
#se non ci sono stampa un messaggio d'errore e termina
if [ $# = 0 ]; then
        log "no args" -1
fi

#mette in TEMP le opzioni da linea di comando
TEMP=‘getopt -o hvw::d:N:m:f:s: -- "$@"‘

#se l'operazione precedente non è andata a buon fine stampa l'errore e termina
if [ $? != 0 ]; then
        log "Parameters parsing error" -1
fi
#mette nei parametri posizionali $TEMP
```

```
eval set -- $TEMP
while [ true ]
do
        #valuta il primo parametro e a seconda se è ammesso un secondo parametro imposta la relativa
        #tramite il comando shift sposta indietro i parametri posizionali
        case $1 in
                -d) DESTINATION_DIR="$2"
                        shift 2
                        ;;
                -N) BACKUP_NAME="$2"
                        shift 2
                        ;;
                -m) case $2 in
                        [0-9]*) BACKUP_NUMBER="$2"
                                if [ $BACKUP_NUMBER -lt 2 ]; then
                                        log "max backups < 2" -1
                                fi
                                shift 2;;
                        *)
                                log "max backups incorrect" -1
                                ;;
                        esac
                        ;;
                -f) EXCLUDE_FILE="$2"
                        shift 2
                        ;;
                -w)
                        NO_MOUNT="true"
                        shift 2
                        ;;
                #l'opzione per i backup remoti imposta SSH_USER e SSH_HOST: l'opzione -s deve essere
                # user@host tramite cut viene diviso questo parametro
                -s) SSH_USER=‘echo $2 | cut -d @ -f 1‘
                        if [ -z $SSH_USER ]
                        then
                                log "ssh syntax incorrect: no user" -1
                        fi
                        SSH_HOST=‘echo $2 | cut -d @ -f 2‘
                        if [ -z $SSH_HOST ]
                        then
                                log "ssh syntax incorrect no host" -1
                        fi
                        SSH_ENABLE="true"
                        shift 2
                        ;;
                -v)
                        VERBOSE="on"
                        shift
                        ;;
                --) shift; break;;
```

```
                -h) echo "
Utilizzo: $0 device mount_point sourcedir [-d destination_dir]
        [-N backup_name] [-m max_backups] [-f eclude_file] [-s utente@host] [-w] [-v]
Opzioni:
 -d destination_dir     La directory in cui salvare il backup
 -N backup_name         Il nome da assegnare al backup
 -m max_backups         Il numero massimo di backup
 -f exclude_file        exclude_file deve esere un file contenete un elenco di file da escludere dal
 -s utente@host         indica a BaTBU di prelevare i dati da un host remoto facendo il login come '
 -w                     forza BaTBU a non montare in ro il file system, (sconsigliato)
 -v                     elenca i valori assegnati tramite le opzioni
        "
        exit;;
                *)
                        log "bad options, type $0 -h for help" -1
                        ;;
        esac
done

#controlla che ci siano almeno 3 parametri
if [ $# != 3 ]
then
        log "Invalid paramters number" -1
fi

#il primo parametro e' il dispositovo da montare
MOUNT_DEVICE=$1
#il secondo è il punto di mount
MOUNT_POINT_RW=$2
#il terzo parametro è la directory sorgente
SOURCE_DIR=$3

if [ "$VERBOSE" = "on" ]
then
        if [ -n "$MOUNT_DEVICE" ]; then
                log "device: $MOUNT_DEVICE" 0
        fi
        if [ -n "$MOUNT_POINT_RW" ]; then
                log "mountpoint: $MOUNT_POINT_RW" 0
        fi
        if [ -n "$SOUCE_DIR" ]; then
                log "source dir: $SOURCE_DIR" 0
        fi
        if [ -n "$DESTINATION_DIR" ]; then
                log "destination dir: $DESTINATION_DIR" 0
        fi
        if [ -n "$BACKUP_NAME" ]; then
                log "backup name: $BACKUP_NAME" 0
        fi
        if [ -n "$BACKUP_NUMBER" ]; then
```

```
                log "max number of backups: $BACKUP_NUMBER" 0
        fi
        if [ -n "$EXCLUDE_FILE" ]; then
                log "exclude file: $EXCLUDE_FILE" 0
        fi
        if [ -n "$NO_MOUNT" ]; then
                log "no mount in ro: $NO_MOUNT" 0
        fi
        if [ -n "$SSH_ENABLE" ]; then
                log "ssh enable: $SSH_ENABLE" 0
                log "ssh user: $SSH_USER" 0
                log "ssh host: $SSH_HOST" 0
        fi
fi

#verifica che l'utente che eseque lo script e' root tramite il comando id -u
#altrimenti notifica l'errore
if [ 'id -u' != 0 ]
then
        log "Must be root user" -1
fi

#controlla che $MOUNT_DEVICE sia un dispositivo a blocchi (un disco)
if [ ! -b $MOUNT_DEVICE ]
then
        log "$MOUNT_DEVICE is not a valid block device" -1
fi

#se $MOUNT_POINT_RW  non e' una directory
if [ ! -d $MOUNT_POINT_RW ]
then
        if [ -e $MOUNT_POINT_RW ]
        then
                #Se $MOUNT_POINT_RW esiste genera un errore grave
                log "$MOUNT_POINT_RW is not a directory" -1
        fi
                #altrimenti genera un warning e crea $MOUNT_POINT_RW
                log "$MOUNT_POINT_RW not exist, making..." 1
        mkdir -p -m 755 $MOUNT_POINT_RW
fi

#se $NO_MOUNT vuota non montare
if [ -z $NO_MOUNT ]
then
        log "mounting $MOUNT_DEVICE on $MOUNT_POINT_RW" 0
        #monta
        mount -o remount,rw $MOUNT_DEVICE $MOUNT_POINT_RW
        #test su valore di ritorno
        if (( $? ))
        then
```

```
                log "can't mount $MOUNT_POINT_RW in rw" -1
        fi
fi

#controlla se $MOUNT_DEVICE è montato in $MOUNT_POINT_RW
if [ -n "$NO_MOUNT" ]
then
        MOUNT_POINT_RW_TEST=`mount |grep $MOUNT_DEVICE |cut -d \  -f 3`
        if [ "$MOUNT_POINT_RW" != "$MOUNT_POINT_RW_TEST" ]
        then
                log "$MOUNT_DEVICE is not mounted on $MOUNT_POINT_DEVICE" -1
        fi
        RW_TEST=`mount |grep $MOUNT_DEVICE | cut -d \  -f6 |cut -d \( -f2 |cut -d \) -f 1`
        if [ "$RW_TEST" != "rw" ]
        then
                log "$MOUNT_DEVICE is not mounted in rw mode" -1
        fi
fi

#controlla che $SOURCE_DIR sia una directory
#if [ ! -d $SOURCE_DIR ]
#then
#       if [ -z $SSH_ENABLE ]
#       then
#               log "$SOURCE_DIR not is a directory" -1
#       fi
#fi

#controlla che non sia abilitato ssh
if [ ! $SSH_ENABLE ]
then
        #controlla che la $SOURCE_DIR esista e sia una directory
        if [ ! -d $SOURCE_DIR ]
        then
                log "$SOURCE_DIR not is a directory" -1
        fi
fi

#se non è stato impostato un nome di backup dai il nome di default "BACKUP"
if [ -z "$BACKUP_NAME" ]
then
        BACKUP_NAME=BACKUP
        log "not -N option specified, assuming \"BACKUP\" name" 1
fi

# se $DESTINATION_DIR non e' una directory
if [ ! -d $MOUNT_POINT_RW/$DESTINATION_DIR/ ]
then
        #se $DESTINATION_DIR esiste
        if [ -e $MOUNT_POINT_RW/$DESTINATION_DIR/ ]
```

```
        then
                #genera un errore fatale
                log "$MOUNT_POINT_RW/$DESTINATION_DIR is not a directory" -1
        fi
                #altrimenti la crea
                log "$MOUNT_POINT_RW/$DESTINATION_DIR not exist, making..." 1
        mkdir -p -m 755 $MOUNT_POINT_RW/$DESTINATION_DIR
fi

#se non è impostato il massimo numero di backup utilizza 5 come default
if [ -z "$BACKUP_NUMBER" ]
then
        BACKUP_NUMBER=5
        log "not -m option specified, assuming 5 for maximum backup" 1
fi

#elimina il backup piu' vercchio
if [ -d $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.$(($BACKUP_NUMBER-1)) ]
then
        log "moving $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.$(($BACKUP_NUMBER-1)) to \
                $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.$(($BACKUP_NUMBER-1)).delete" 0

        mv $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.$(($BACKUP_NUMBER-1)) \
                $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.delete

        log "deleting $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.delete" 0

        rm -rf $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.delete &
fi

#shifto i backup da 2 in poi di una posizione
for i in 'seq 2 $BACKUP_NUMBER |sort -r'
do
        #echo $i
        #if [ $i -eq $BACKUP_NUMBER ]
        #then
        #       break
        #echo "pipp"
        #fi
        iminus=$(($i-1))
        if [ -d $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.$iminus ]
        then
                log "rotating $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.$iminus to \
                $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.$i" 0
                mv $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.$iminus $MOUNT_POINT_RW/$DESTINATIO
        fi
done

#creo degli hard link ricorsivamente al backup piu' recente in $MOUNT_POINT_RW/$DESTINATION_DIR/$BAC
if [ -d $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.0 ]
```

```
then
        log "copying $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.0 to \
        $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME."0
        cp -al $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.0 $MOUNT_POINT_RW/$DESTINATION_DIR/$BAC
fi

#tengo conto dei file da escludere
if [ ! -z $EXCLUDE_FILE ]
then
        #se sono impostati dei file da escludere crea l'opzione per rsync
        if [ -f $EXCLUDE_FILE ]
        then
                EXCLUDE_LINE="--exclude-from=\"$EXCLUDE_FILE\""
        else
                log "$EXCLUDE_FILE is not a valid file" 1
        fi
fi

#se ssh e' abilitato crea l'opzione da passare come parametro a rsync
if [ $SSH_ENABLE ]
then
        SSH_LINE="-e ssh $SSH_USER@$SSH_HOST:"
fi

log "synking $SSH_LINE$SOURCE_DIR/ to $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.0 excluding $EXC

#sincronizza $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.0 con la sorgente
#copiando solo le differenze

rsync -va --delete-excluded $EXCLUDE_LINE $SSH_LINE$SOURCE_DIR/ $MOUNT_POINT_RW/$DESTINATION_DIR/$BA

#imposta la data di modifica alla data corrente
touch $MOUNT_POINT_RW/$DESTINATION_DIR/$BACKUP_NAME.0

wait

#se non è specificato rimonta in ro
if [ -z $NO_MOUNT ]
then
        log "mounting ro $MOUNT_DEVICE to $MOUNT_POINT_RW" 0
        mount -o remount,ro $MOUNT_DEVICE $MOUNT_POINT_RW
        if (( $? ))
        then
                log "Error when mounting ro $MOUNT_POINT_RW" -1
        fi
fi

log "Backup done." 0
```

8.1 Appendice A: autenicazione SSH tramite chiave RSA

Per evitare l'interazione con l'utente nella fase di autenticazione sulla macchina remota occorre configurare Open SSH per l'autenticazione con chiavi crittografiche.

La procedura passo passo:

- Generare una coppia RSA: **`ssh-keygen -t rsa`**
- In questo modo vengoni generati due file: **`id_rsa`** e **`id_rsa.pub`** che contengono ciascuno la chiave privata e pubblica
- Sull'host remoto creare nella directory **`/root/.ssh/`** il file **`authorized_keys2`**
- Inserire la chiave pubblica nel file (in append) **`cat id_rsa.pub >> authorized_keys2`**