Clupdate: uno strumento per l’aggiornamento del cluster Arcosu

Carlo Podda e Alan Scheinine, Area HPC, CRS4

CRS4
Centro di Ricerca, Sviluppo e Studi Superiori in Sardegna
Polaris, Edificio 1
Loc. Pixina Manna
09010 Pula (Cagliari), Italia

E-mail: carlo@crs4.it, scheinin@crs4.it

Sommario

1 Introduzione

Ogni computer nel cluster di calcolo del CRS4, "il cluster Arcosu", ha dei file configurati dall'area High Performance Computing (HPC) subito dopo aver installato il sistema operativo. Tra questi sono compresi file di configurazione, come per esempio `ntp.conf`, `nsswitch.conf`, `arcosu.conf` e `sendmail.mc`, e file di dati come `hosts`, `exports` e `auto.mnt`. Alcuni sono uguali per tutte le macchine del cluster (host), ad es. `/etc/hosts`; altri sono uguali per una particolare versione del sistema operativo, ad es. `/etc/mail/sendmail.mc`; altri ancora sono specifici per una macchina, ad es. `/etc/exports`. Per semplificare le operazioni di gestione di gruppi di host, e per mantenere traccia dei cambiamenti effettuati, è stato creato, in una cartella esportata tramite NFS, ($CLUPDATE_ROOT), un archivio contenente tutti questi file. Quindi, per permettere l'aggiornamento degli host con i file presenti nell'archivio, è stato creato il programma Clupdate.

Durante l'installazione del sistema operativo dalla rete, fatta con PXE boot e kickstart, è anche possibile modificare tutti i file di configurazione. La scrittura di questi file da kickstart (ks.cfg) non è però conveniente, questo perché lo script ks.cfg diventa troppo grande e comunque il contenuto dei file di configurazione aggiornati rimane fuori da un archivio. Un metodo migliore è quello di inizializzare la configurazione del sistema operativo abbastanza da poter poi consentire un successivo aggiornamento con Clupdate. Per utilizzare Clupdate è sufficiente attivare la connessione di rete, configurare ssh e il client NFS.

$CLUPDATE_ROOT, con le sue sottodirectory, rispecchia la struttura stessa del cluster. Ogni host è presente o con una sua sottodirectory oppure in quanto appartenente a una categoria (gruppo di host) con una propria sottodirectory. Ad esempio:

```
$CLUPDATE_ROOT/nuradha/, root della gerarchia nuradha,
 un host specifico
$CLUPDATE_ROOT/nuraxi1/, root della gerarchia nuraxi1,
 un host specifico
$CLUPDATE_ROOT/nuraxi/, root della gerarchia nuraxi,
 un gruppo di nodi  (nuraxi1,...,nuraxi8)
$CLUPDATE_ROOT/all_hosts/, root gerarchia all_hosts,
tutti i nodi (nuradha, nuraxi,...)
```

Da notare come un host può apparire più volte: con una sua specifica cartella ma anche come appartenente ad una o più categorie.

Nell'aggiornare un host, un file all'interno di una gerarchia dal nome più specifico ha la precedenza rispetto allo stesso file presente in una gerarchia dal nome più generale. Non è necessario avere un'esplicita gerarchia per ogni host. Per fare degli esempi concreti la gerarchia di nuraxi1 è creata solo se nuraxi1 ha, supponiamo, un suo file system da esportare in NFS; quindi rispetto alle altre nuraxi ha bisogno che esista il suo file /etc/exports. Comunque non è necessario che ogni file appaia in ogni gerarchia. L'idea alla base di Clupdate è quella di minimizzare la quantità di lavoro necessario per specificare i casi generali ed i casi particolari.

Clupdate è composto da diverse parti:
lo script `$CLUPDATE_ROOT/bin/clupdate.pl` è il driver;
il modulo `$CLUPDATE_ROOT/bin/Arcosu/Gerarchia.pm` contiene le funzioni per navigare nella gerarchie;
un altro script, `$CLUPDATE_ROOT/bin/chkpath.pl`, fa un controllo sulla presenza di link nel path del host target. Questi link infatti potrebbero "ingannare" il programma che si preoccupa di trasferire i file, `rsync`;
infine, il file `$CLUPDATE_ROOT/bin/macchine` descrive le gerarchie delle cartelle associate ai nodi o alle categorie.

Clupdate si utilizza specificando la destinazione (come un host o una categoria di host), un file da trasferire (o una cartella o un "glob", cioè "path" con "wild cards") e delle opzioni di funzionamento (copia ricorsiva dei file, crea backup dei file sostituiti etc).

L'algoritmo è composto da più fasi; la prima fase consiste nel trasformare la destinazione in un elenco di nodi, nel caso in cui questa fosse specificata come una categoria. Questa operazione è fatta con una chiamata alle funzioni di gerarchia.pm. Quindi clupdate esegue un controllo dei file, cartelle o glob da trasferire. Verifica che facciano parte di una determinata lista di file o cartelle, in altre parole che siano contenuti sotto $CLUPDATE_ROOT/ e che non contengano parti "insolite" nel loro nome o path (per esempio "../"). Subito dopo, per ogni host e per tutti i livelli della gerarchia, viene fatta una raccolta dei corrispondenti file, cartelle o glob da trasferire.
Per esempio per l'host nuraxi1 la ricerca viene fatta nelle gerarchie: `all_hosts`, `nuraxi` e infine `nuraxi1`. Quindi, con chkpath.pl, viene analizzato il path del target host dove verranno copiati i file. In particolare controlla se nel target path esistono o no dei link simbolici e se l'intero path è presente o se, rispetto al source path, ne esiste solo una parte. Sulla base dei file da trasferire e sul controllo di chkpath verranno generati i comandi di rsync per effettuare la copia.

Per minimizzare il numero di comandi di `rsync`, i file sono raggruppati secondo l'host o la cartella di destinazione.

Il capitolo 2 descrive il programma principale: clupdate.pl. Il capitolo 3 descrive un programma per controllare il target path: chkpath.pl. Il capitolo 4 descrive un modulo di Perl con le funzioni per trattare di una gerarchia di sorgenti: Gerarchia.pm.

2 clupdate.pl

Perl script per aggiornare i file del cluster

SYNOPSIS

```
clupdate.pl -{t|T} {host target|class target}
   {-d|-D|-f} {file or directory}  -b -r -i -n --debug

-t, -T, -target, -TARGET  hostname|classname
                        Host o classe di host di destinazione.

-d, -D, -f, -F  ‘‘file-o-cartella1 file-o-cartella2 ...
... file-o-cartellaN’’
                        Nessuna distinzione viene fatta tra
                        file e cartelle con questa opzione.

-help, -h, --help, --h  Stampa a video questa pagina

-b, -B                  Esegue un backup dei file sostituiti.
                        Il backup e‘ fatto nello stesso file
                        system che contiene l’archivio. Opzionale

-i, -I                  Interattivo. Opzionale.

-n, -N                  Fa una prova senza eseguire la copia.
Opzionale.

-all                    Copia tutto il contenuto dell’archivio.
Opzionale.


--debug                 Attiva debug. Opzionale.

-r, -R                  Copia ricorsiva del contenuto delle cartelle.
Opzionale.
```

Descrizione

Clupdate è stato creato per semplificare le operazioni di gestione e di configurazione delle macchine di un cluster. In un cluster per il calcolo, come il nostro "Arcosu", le installazioni degli host sono sostanzialmente tutte uguali; l'eccezione riguarda pochi file (per fare un esempio, quelli che contengono il nome o l'indirizzo IP della macchina o la configurazione del server NFS). La modifica di file comuni a tutte le macchine risulta estremamente ripetitiva. Per eliminare la duplicazione dei comandi abbiamo creato un repository che ripetesse, per ogni macchina o gruppo di macchine, la struttura del sistema operativo. Questo repository è il luogo dove mettere tutti i file di sistema da personalizzare o quelli nuovi da creare. Una volta fatte le modifiche o le aggiunte ai file o directory dentro il repository, con l'aiuto di clupdate sarà possibile con una sola riga di comando copiarli su tutti i nodi. Oltre alla possibilità di aggiornare su diversi host con un unico comando, l'utilità di questa soluzione è anche quella di avere un "luogo" dove si mantiene traccia di quali file o directory sono stati aggiornati o aggiunti.

Il repository è organizzato in modo da avere differenti "directory-gerarchie", per distinguere i diversi gruppi di macchine. A partire da queste "directory-gerarchie", il programma puó copiare tutti i file presenti a qualsiasi livello di sottodirectory oppure fare una copia mirata di un singolo file o di una intera struttura di directory con tutti i file all'interno. Il file da copiare, tramite rsync, viene confrontato con il corrispondente sulla macchina remota. Se ci sono differenze viene copiato altrimenti la copia non viene fatta.

Clupdate legge la lista dei nodi, e i loro eventuali raggruppamenti logici, da un file di testo: il file "$CLUPDATE_ROOT/bin/macchine". A partire da questo file viene creato, su un hash, un albero con la radice denominata "all_hosts". La radice può avere ha come suoi figli raggruppamenti di nodi (es. tutte le nuraxi, tutte le scivu, tutti i server) o direttamente i nomi delle macchine. Questi nodi dell'albero che sono raggruppamenti di host possono avere a loro volta come figli gli host specifici o altri raggruppamenti. Il tutto può essere implementato su un numero di livelli in teoria illimitato. Per ogni foglia, a cui corrisponde un singolo host, può esistere o no una directory sotto $CLUPDATE_ROOT. Se tale directory non esiste, o se non contiene file, si procede al controllo sulla directory cui fa riferimento il nodo padre, e così via sino a trovare i file che ci interessa copiare.

Se scorrendo l'albero vengono trovate più copie di un file viene utilizzata quella ospitata nella directory più in basso, cioè la copia nella directory con il nome dell'host oppure, scorrendo l'albero verso l'alto, quella più vicina a questa.

Alcuni esempi di utilizzo di clupdate con i comandi di rsync generati:

```
clupdate.pl -t nuraxi1 -d /etc/
```

Clupdate crea i comandi per fare una copia, sull'host nuraxi1, del file /etc/resolv.conf. Questo file è cercato in: $CLUPDATE_ROOT/all_hosts/etc, $CLUPDATE_ROOT/nuraxi/etc $CLUPDATE_ROOT/nuraxi1/etc.

Il comando generato è

```
cd $CLUPDATE_ROOT//all_hosts/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc/resolv.conf \
nuraxi1:/
```

Volendo invece copiare tutti i file di /etc su tutte le nuraxi:

```
clupdate.pl -t nuraxi -d /etc/
```

I file da copiare sono cercati in: $CLUPDATE_ROOT/all_hosts/etc/ $CLUPDATE_ROOT/nuraxi/etc/ $CLUPDATE_ROOT/nuraxi1/etc/ ... ... $CLUPDATE_ROOT/nuraxi8/etc/

Il comando generato è:

```
cd $CLUPDATE_ROOT//all_hosts/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc//resolv.conf etc//hosts \
  nuraxi1:/
cd $CLUPDATE_ROOT//nuraxi/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc//auto.mnt nuraxi1:/
cd $CLUPDATE_ROOT//all_hosts/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc//resolv.conf etc//hosts \
```

```
 nuraxi2:/
...
...
cd $CLUPDATE_ROOT//all_hosts/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc//resolv.conf etc//hosts \
 nuraxi8:/
cd $CLUPDATE_ROOT//nuraxi/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc//auto.mnt nuraxi8:/
```

Copia ricorsiva dei file e cartelle sotto /etc, su nuraxi1

```
clupdate.pl -t nuraxi1 -d /etc/ -r
```

I file da copiare, in questo caso sono in:
$CLUPDATE_ROOT/all_hosts/etc/
$CLUPDATE_ROOT/nuraxi/etc/
$CLUPDATE_ROOT/nuraxiX/etc/
e in tutte le sottocartelle presenti.

Il comando generato è:

```
cd $CLUPDATE_ROOT/all_hosts/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc/hosts etc/resolv.conf \
  nuraxi1:/
cd $CLUPDATE_ROOT/nuraxi/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc/auto.mnt nuraxi1:/
cd $CLUPDATE_ROOT/all_hosts/
/usr/bin/rsync -vzlptgoD -e /usr/bin/ssh  etc/init.d/autofs \
etc/init.d/bbd nuraxi1:/etc/rc.d/init.d/
cd $CLUPDATE_ROOT/nuraxi/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc/mail/sendmail.mc nuraxi1:/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc/rc.d/init.d/autofs nuraxi1:/
cd $CLUPDATE_ROOT/all_hosts/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc/rc.d/rc3.d/S95bbd nuraxi1:/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc/rc.d/rc4.d/S95bbd nuraxi1:/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  etc/rc.d/rc5.d/S95bbd nuraxi1:/
```

Da notare come le opzioni di rsync cambiano a seconda del file da copiare.

Infine la copia del file /etc/auto.mnt su nuraxi1, con salvataggio del file sostituito.

```
clupdate.pl -t nuraxi1 -f /etc/auto.mnt -b
```

Il file sostituito su nuraxi1 verrà copiato dentro una cartella, creata un attimo prima, con un nome e un path significativi.

Il comando generato è:

```
mkdir $CLUPDATE_ROOT/Backup/nuraxi1/06-09-2005_11-32-06
cd $CLUPDATE_ROOT//nuraxi/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh -b  --backup-dir=\
$CLUPDATE_ROOT//Backup/nuraxi1/06-09-2005_11-32-06 \
etc/auto.mnt nuraxi1:/
```

Dettagli sull'implementazione

Clupdate cambia a seconda dell'occorrenza le opzioni di rsync. L'insieme completo di opzioni che possono essere utilizzate è:
"-vzlptgoDR"

Le opzioni di rsync: v=verbose, z=compress file data during send,l=copy symlinks as symlinks, p=preserve permissions, t=preserve times, g=preserve group, o=preserve owner, D=preserve devices, R=use relative path names.

In particolare interessa l'opzione "-R", utile per trasferire un gruppo di cartelle. Dai manuali di rsync:

```
 -R, --relative
       Use  relative  paths. This means that the full path names speci-
       fied on the command line are sent to the server rather than just
       the  last  parts  of  the filenames. This is particularly useful
       when you want to send several different directories at the  same
       time. For example, if you used the command

       rsync foo/bar/foo.c remote:/tmp/

       then  this  would  create  a  file  called foo.c in /tmp/ on the
       remote machine. If instead you used

       rsync -R foo/bar/foo.c remote:/tmp/
```

```
then a file called /tmp/foo/bar/foo.c would be created on the
remote machine. The full path name is preserved.
```

Il comportamento di clupdate, e conseguentemente le opzioni di rsync generate, sono condizionate da come si presenta il path di destinazione.
Due sono le condizioni da verificare: se nel path di destinazione ci sono link simbolici e se il path esiste interamente o solo in parte. Queste condizioni sono controllate da uno script in Perl invocato da clupdate: lo script chkpath.pl.
Chkpath.pl è in grado di distinguere se il path di destinazione presenta dei link simbolici e se esiste tutto solo in parte. Questo script è descritto nel capitolo 3.

In base alla classificazione del path di destinazione distinguiamo tra quattro casi diversi.

Il path di destinazione del target è completo

Nessun link simbolico è presente nel path di destinazione

Questo è il caso più semplice da trattare. Clupdate non fa nessuna operazione particolare nel path di destinazione, rsync viene eseguito con tutte le opzioni: "-vzlptgoDR".

Il comando di rsync generato avrà come argomenti:
source file= il file da copiare completo di path
(a partire da $CLUPDATE_ROOT/gerarchia)
target file= il nome dell'host remoto senza nessun path.

Una o più directory del path di destinazione sono link simbolici.

Talvolta un determinato path ha un nome convenzionale sotto forma di link simbolico. Questo link simbolico punta alla directory che contiene l'installazione vera e propria, questa presenta spesso nel suo nome anche una serie di numeri o lettere ad indicare versione, upgrade etc. Per esempio, per il software Big Brother (BB) il nome della cartella con il software comprende anche numeri e lettere; a questa cartella viene fatto puntare un link simbolico dal nome più

generale (bb). Questo link è richiesto per il funzionamento BB oltre che per semplificarne la gestione.

Nel nostro repository, per esempio, è presente il file di testo

```
$CLUPDATE_ROOT/all_hosts/home/bbuser/bb/etc/bb-hosts
```

In questo file BB conserva l'elenco di tutti i client e server. È comodo avere un solo file come questo da aggiornare quando ci sono modifiche nella lista delle macchine di BB.

Di solito BB si installa a partire dalla home dell'utente bbuser. Qui esiste un link simbolico "bb " che fa riferimento alla directory con l'installazione vera e propria.
/home/bbuser/bb → /home/bbuser/bb.$BB_VER

In tutti i nodi e i server di BB esiste un path /home/bbuser/bb, non in tutti i nodi però, è necessariamente installata la stessa versione del software.
$BB_VER può cambiare da un host all'altro.

Con il comando:
clupdate.pl -t all_hosts -f /home/bbuser/bb/etc/

Utilizzando rsync con le opzioni descritte per il caso precedente, e senza eseguire un controllo sui link simbolici, sulla macchina di destinazione verrebbe creata una cartella con il nome "bb/" cancellando il link simbolico; in questa cartella verrebbe creata la dir "etc/" e dentro questa copiato il file bb-hosts. L'installazione di BB, (/home/bbuser/bb.$BB_VER), non sarebbe aggiornata.

Per superare questo problema clupdate "modifica" il path di destinazione di rsync, eliminando il nome dei link simbolici e mettendo al loro posto la directory cui puntano.

Ad esempio il comando:
./clupdate.pl -t nuradha -d /home/ -r

genera le seguenti righe di comando:

```
cd $CLUPDATE_ROOT/all_hosts/
/usr/bin/rsync -vzlptgoD -e /usr/bin/ssh home/bbuser/bb/etc/bb-hosts \
nuradha://home/bbuser/bb19c/etc/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  home/bbuser/dir1/dir2 nuradha:/
```

N.B. le opzioni di rsync sono modificate solo quando serve.

Rsync viene eseguito con tutte le opzioni "-vzlptgoD", ed ha come argomenti:
source file= il file da copiare completo di path
(a partire da $CLUPDATE_ROOT/gerarchia) target file= nome dell'host remoto con path completo e "modificato".

Il path di destinazione del target non è completo.

Nessun link simbolico è presente nel path di destinazione

In questo caso il programma utilizza le opzioni "-vzlptgoDR" per creare le directory che non sono presenti nel target.

```
clupdate.pl -t nuradha -d /home/prova/dir_non_nel_target -r

cd $CLUPDATE_ROOT//nuradha/
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  home/non_nel_target/file2copy \
nuradha:/
```

Rsync viene eseguito con tutte le opzioni "-vzlptgoDR", ed ha come argomenti:
source file= il file da copiare completo di path
(a partire da $CLUPDATE_ROOT/gerarchia)
target file= nome dell'host remoto senza alcun path di destinazione.

Ci sono link simbolici nel target path.

L'ultimo caso è quello in cui il path di destinazione esiste solo in parte e presenta al suo interno dei link simbolici. In questo caso le opzione di rsync saranno nuovamente:
"-vzlptgoDR"

```
clupdate.pl -t nuradha -d /home/prova/link2dir1/non_nel_target -r
cd $CLUPDATE_ROOT/nuradha//home/prova/link2dir1/
```

```
/usr/bin/rsync -vzlptgoDR -e /usr/bin/ssh  non_nel_target/file \
nuradha://home/prova/dir1/
```

Nell'esempio precedente clupdate deve copiare tutti i file e le directory presenti sotto /$CLUPDATE_ROOT/nuradha/home/prova/link2dir1 nell'host nuradha. Nuradha ha il path di destinazione sino a link2dir1 (che però è un link alla cartella /home/prova/dir1). La cartella "non_nel_target" invece non esiste su nuradha.

Rsync viene eseguito con tutte le opzioni -vzlptgoR. Il programma si posiziona nell'ultima cartella presente sia nel source che nel target (/home/prova/link2dir1) e rsync ha come argomenti:
source file= il file da copiare completo di path relativo (da dove ci troviamo).
target file= nome dell'host remoto con path "modificato" comune ai due host.

AUTHOR

Carlo Podda, carlo@crs4.it

COPYRIGHT AND LICENSE

3 chkpath.pl

Perl script per adeguare il path di clupdate all'utilizzo di rsync

SYNOPSIS

```
chkpath.pl nodo-del-cluster path-2-check
```

Descrizione

Con clupdate, oltre l'aggiornamento di uno o più file di configurazione, può verificarsi la necessità di copiare o creare un'intera gerarchia di cartelle. In questo caso l'uso dell'opzione "`-R`" di rsync è conveniente. Per esempio, il comando
`rsync foo/bar/foo.c remote:/tmp/`
crea il file `foo.c` nella cartella `/tmp/`.

Invece, il commando
`rsync -R foo/bar/foo.c remote:/tmp/`
crea il file `/tmp/foo/bar/foo.c`.

Questo è importante quando il fine è quello di trasferire una gerarchia di file anzichè un solo file. Il problema che sorge in questo caso è il trattamento dei link simbolici nella destinazione; infatti con l'utilizzo dell'opzione "`-R`" rsync sostituisce il link simbolico con una cartella con lo stesso nome. Il commando `rsync` con questa opzione, ricrea l'intera struttura da copiare sino al file finale, anche se una parte delle directory già esistevano nel'host target sotto forma di link.

Chkpath.pl è stato creato per superare questo problema che si può avere facendo una copia ricorsiva.

Il programma chkpath.pl riceve come parametri in input il nome di un host ed un path da controllare e verificare.

Il path in input viene controllato; in output viene restituito un altro path in cui i link, se esistono, sono sostituiti della directory cui puntano. Questo permette

a clupdate.pl di creare il comando di rsync specificando esattamente il path di destinazione. Se un path esiste solo parzialmente sul target il programma lo segnala ugualmente, restituendo in output solo la parte di path esistente.

Il programma, avviato con le opzioni

```
/usr/bin/ssh chkpath.pl hostname path-2-check
```

restituisce un output del tipo

```
{ false | true | broken } N target-part source-path [ diff ]
```

Il primo campo segnala se nel path ci sono link (`true`), se non ce ne sono (`false`) o se ci sono (`broken` link), cioè link che puntano a una cartella che non esiste.
Il secondo campo (`N`) è un numero intero che riporta il numero di directory esistenti nel target, sia come directory che come link simbolici.
Il terzo campo, (`target-path`), indica il path nel target host (sino a dove esso esiste), sostituendo i nomii delle cartelle al posto dei link simbolici.
Il quarto campo, (`source-path`), indica qual'è, sull'host sorgente, la parte di path che ha una corrispondente anche nel target.
Il quinto campo, (`diff`), è la parte del path che esiste sul source ma non sul target.

L'output di chkpath viene elaborato da clupdate.pl. Sulla base di questo il programma principale crea i comandi rsync appropriati.

AUTHOR

Carlo Podda, carlo@crs4.it

COPYRIGHT AND LICENSE

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself.

4 Gerarchia

Perl module to deal with hierarchy of machine files.

SYNOPSIS

```
use Arcosu;


# Example definition of $gerarchia_file before call to get_gerarchia.
$cluster_bin = '/CLUSTER/share/bin/Clupdate' ;
$gerarchia_file = '/CLUSTER/share/bin/Clupdate/macchine' ;


# Must be called first.
# Creates the hash of hashes (tree) %Arcosu::Gerarchia::gerarchia
# and creates the hash of lists %Arcosu::Gerarchia::gerarch_gruppi
&get_gerarchia($gerarchia_file);


# Print the tree, a program for testing.
# Implicitly uses %Arcosu::Gerarchia::gerarchia
# Equal to print_tree_specific(\%Arcosu::Gerarchia::gerarchia);
&print_tree();


# Print the tree, a program for testing.
# Could use a hash different from %Arcosu::Gerarchia::gerarchia
&print_tree_specific(\%any_gerarchia);


# Example, getting a list of leaves.
# print "Leaves:\n";
# @leaf_list = &list_leaves(\%gerarchia);
# foreach $leaf (@leaf_list) {
#     print "$leaf\n";
# }
# Equal to list_leaves_specific(\%Arcosu::Gerarchia::gerarchia);
@leaf_list = &list_leaves();
```

```
# Gets a list of leaves, that is, computers.
# The call parameter can be a hash different from
# %Arcosu::Gerarchia::gerarchia , in particular,
# it can be a subtree of the hierarcy.
@leaf_list = &list_leaves_specific(\%any_gerarchia);


# Given a machine name, such as $dest = cixerri2 or
# given a category name, such as $dest = nuraxi ,
# returns a list of the corresponding computers.
# That is, returns the leaves of the tree of categories,
# where the leaves are actual computers.
# Uses the internal function get_subtree() to get the
# subtree hierarcy that starts with $nodename.
@destination_nodes = &get_dest_nodes($dest);


# Given a machine name, such as $dest = cixerri2 or
# given a category name, such as $dest = nuraxi ,
# return nodes of tree in path from $dest to root of tree.
# That is, the hierarchy from the given category, $dest,
# to the most general category.
@above_list = &get_above($dest);


# Given a file path, such as $filepath = /etc/hosts , and
# given a category of computers such as
#     $category = nuraxi  (a group of computers)
# or  $category = cixerri2  (one specific computer)
# and given the directory that has the
#  machine directories, $cartelle_share ,
# returns a hash whose keys are specific machine names
# and the values of the hash are references to arrays
# that contain a list of files.
# Each file list has as the first element (index 0) the file
# that has precedence for being sent to the machine,
# the other files in the list can be ignored, they are
# included in order to be informative.
%macchine_e_file =
   &nodes_and_file($filepath, $category, $cartelle_share);


# Given a f directory, such as $dirpath = /etc , and
# given a category of computers such as
```

```
#   $category = nuraxi  or  $category = cixerri2
# and given the directory that has the machine directories,
# returns a hash whose keys are specific machine names
# and the values of the hash are references to hashes,
# the second level of hashes has as keys the files found
# in the directory $dirpath, and as values, references to arrays
# that contain a list of files.
# Each file list has as the first element (index 0) the file
# that has precedence for being sent to the machine,
# the other files in the list can be ignored, they are
# included in order to be informative.
# The algorithm employed is that for a given category, a list of
# machines is created, then for each machine an ordered list
# of categories is created, from the specific machine to all_hosts.
# For each specific machine, the given $dirpath is search using
# "find" to get a list of files in any directory in the
# ordered list of categories.  Then for each file (for a specific
# machine) the algorithm in the routine nodes_and_file is used
# to determine which source file has precedence.
# An example of creating a set of rsync commands is shown below.
# The first level of hash has as keys the destination machines
# and destination directory.
# Notice that the keys at the second level of hash are not needed,
# they are informative as to which file is being considered,
# but in practice only the values are needed.  With regard
# to the second level of hash, each value is a reference to
# a list, of which only the first element is needed.
# The other elements are source files with lower priority.
%macchine_e_dir =
   &nodes_and_directory($dirpath, $category, $cartelle_share);
@all_destinations = keys(%macchine_e_dir);
@pretty = sort @all_destinations;
foreach $dest (@pretty) {
   print "\n/usr/bin/rsync -avI --rsh=/usr/bin/ssh " . " \\" . "\n";
   %files = %{$macchine_e_dir{$dest}};
   foreach $afile (keys(%files)) {
      @possible = @{$files{$afile}};
      print " $cartelle_share" . '/' . $possible[0] . " \\" . "\n";
   }
   print "   ${dest}:$dirpath/\n";
}
```

```
# Same as subroutine nodes_and_directory except the path
# can be a directory, a file, or a path with wild cards.
# If the recursive option is not specified, then a glob
# that translates into a directory will refer only to the
# files in that directory, whereas with the recursive option
# the behavour is due to a glob that translates into a directory
# is the same as the routine nodes_and_directory().
%macchine_e_glob =
   &nodes_and_glob($globpath, $category, $cartelle_share,
                   [ recursive | ricorsivo] );
```

ABSTRACT

Given a file that describes the hierarchy of precedence between computer names and category names, and give a directory in which each subdirectory corresponds to a computer or a catergory (a set of computers), and assuming that the hierarchy has a tree structure, manipulates information concerning the hierarchy and concerning the files found in the subdirectories.

DESCRIPTION

Overview

A specific file, for example a configuration file for a specific service of the operating system, may be appropriate for one specific computer (for example, NFS exports of a specific file server) or may be appropriate for a group of computers (for example, the ethernet configuration for all nodes of a specific sub-cluster with a specific type of mainboard). Various directory hierarchies in an archive have been created to organize these files. A single hierarchy has as the top directory a name such as "scivu1", "scivu" or "all_hosts" and under each such directory is a hierarchy of directories as would appear on the target computer, that is, directories such as /etc/, /home/bbuser/ and so on. If a file appears in the "scivu1" hierarchy then it should be installed on the computer named scivu1. If it appears in the "scivu" hierarchy then it should be installed on all computers scivu1 to scivu8. If it appears in "all_hosts" then the file is common to all computers of the cluster. There is a file (assigned to the variable $gerarchia_file) that specifies the hierarchial relationship of the top-level directories. If a file with the same name appears in two hierarchies, then the more specific hierarchy takes

precedence in the updating of the files from the archive to the target computers. As an example, it is convenient to have a local file /etc/hosts that gives the IP addresses of computers on the cluster. In addition, the nodes nora01 to nora16 have been used for testing SRB file system clients using the second Gigabit Ethernet interface (NIC), so under the hierarchy that begins with the directory "nora" there is an /etc/hosts file that includes the IP addresses relative to the second NIC. In this way, it is not necessary to put the special "hosts" file under sixteen different hierarchies, nora01 to nora16, instead just one copy is needed under the hierarchy "nora".

The programs in the package Arcosu::Gerarchia can deal with relatively complex cases, such as listing the files that need to be transfered when a partially-specific hierarchy is given and a directory or even a "glob" (that may specify many directories) is given as input. As a specific case, consider the partially specific hierarchy (formally a 'category') called "nuraxi" as a given input. If one specific file is to be transfered then the logic is that "nuraxi" specifies the set of computers nuraxi1 to nuraxi8. For each computer, the specific file can come from any one of several different hierarchies. Consider the computer nuraxi3. If there exists a hierarchy of directories that begins with a directory called "nuraxi3" and if the given file (taken as the full path relative to '/') is found under "nuraxi3" then that file is transfered; but if the file is not found under "nuraxi3" but is found under the "nuraxi" hierarchy, then the file under "nuraxi" is transfered; or if the file is not under "nuraxi" but is under "all_hosts" then the file is taken from the latter hierarchy. If the user asks to transfer a directory or a glob (the use of wild-card characters), then the logic is the same as for an individual file, after the directory or glob is translated into a collection of files. For example, if the "nuraxi" hierarchy (category) is given, then this is translated into nuraxi1 to nuraxi8 and for each leaf node the program looks at the most specific to the most general (e.g. "nuraxi3" then "nuraxi" then "all_hosts") making a list of all files that correspond to the directory or glob. Then for each file in this list of files, the previously mentioned procedure is followed in order to select which file takes precedence.

Though the procedure sounds complicated, it has been synthesized into just a small number of Perl routines. Moreover, in practice the concept is simple. The specification of a directory to update, rather than a file, is appropriate for the case of updating a software package such as Big Brother in /home/bbuser. Moreover, if just one computer, e.g. nuraxi3, needed to have just one file of the Big Brother software different from the general case, then the user need only put that file in, for example, "nuraxi3/home/bbuser/bb/etc/bb-proctab". Moreover, if there is one general configuration of bbuser for all computers and the hierarchy (category) specified is "nuraxi", it is not necessary that the files of /home/bbuser be under the hierarchy "nuraxi" if they are under the hierarchy "all_hosts"; specify-

ing "nuraxi" as the category in the command means that only nuraxi1 to nuraxi8 will be updated.

Export

Subroutines &get_gerarchia &print_tree &print_tree_specific &list_leaves &list_leaves_speci &get_dest_nodes &get_above &nodes_and_file &nodes_and_directory

Variables %gerarchia %gerarch_gruppi

Exportable constants

```
   $Arcosu::Gerarchia::VERSION
For example
   print "Arcosu::Gerarchia version = $Arcosu::Gerarchia::VERSION\n";
```

AUTHOR

Alan Louis Scheinine, scheinin@crs4.it

COPYRIGHT AND LICENSE

Ringraziamenti

Questo lavoro è finanziato dal MUIR D.D. 9/10/2002 prot. 1105/2002 Rif. prog. n. 212 e dalla Regione Autonima della Sardegna.