Ganeshell: browsing/debugging tool for GANESHA NFS server

thomas.leibovici@cea.fr

13th May 2008

Contents

1 Getting started

1.1 Compiling, installing and running

Build the NFS-GANESHA distribution with the filesystem that you want.
E.g:

```
./configure --with-fsal=<filesystem_type>
make
```

In the "shell" directory, a program called `<fsname>.ganeshell` is built. You can install it on your system by running "`make install`".

Then, you can simply run the shell without arguments (interactive mode), or specify one or several script files to be executed (batch mode).

```
Synopsis:
<fsname>.ganeshell [-h][-v][-n <nb>][Script_File1 [Script_File2]...]

-h: display short help
-v: verbose mode
-n <nb>: number of script instances (threads) to run
```

1.2 Start an interactive session

Run the shell with no arguments, then type "help" for displaying available commands:

```
bash$ ./posix.ganeshell

ganeshell>help
Shell built-in commands:
           barrier: synchronization in a multi-thread shell
              echo: print one or more arguments
              exit: exit this shell
              help: print this help
                if: conditionnal execution
       interactive: close script file and start interactive mode
             print: print one or more arguments
              quit: exit this shell
               set: set the value of a shell variable
              time: measures the time for executing a command
             unset: free a shell variable
           varlist: print the list of shell variables

Shell tools commands:
             chomp: removes final newline character
               cmp: compares two expressions
```

```
            diff: lists differences between two expressions
              eq: test if two expressions are equal
         meminfo: prints information about memory use
              ne: test if two expressions are different
           shell: executes a real shell command
           sleep: suspends script execution for some time
           timer: timer management command
              wc: counts the number of char/words/lines in a string

Layers list:
            FSAL: File system abstraction layer
     Cache_inode: Cache inode layer
             NFS: NFSv2, NFSv3, MNTv1, MNTv3 protocols (direct calls, not through RPCs)
      NFS_remote: NFSv2, NFSv3, MNTv1, MNTv3 protocols (calls through RPCs)
```

2 Using the shell

2.1 Layers

2.1.1 Layers overview

The shell deals with GANESHA's layers independently. You must initialize each layer that you want to use in the correct order (first the lowest layer, then the upper ones), and you can issue some commands on each of these layers.

The list of available layers is displayed in the return of the "help" command.

Here is the list of server-side layers, from the lowest level to the highest:

- FSAL: the File System Abstraction Layer
- Cache_inode: the metadata cache layer (FSAL layer must be initialized)
- NFS: the NFS protocol layer (FSAL and Cache_inode layers must be initialized)

You can also use ganeshell only as a NFS client, with the "NFS_remote" layer. This layer doesn't need any other layer to be initialized.

2.1.2 Setting current layer

You can set the current layer by setting the `LAYER` shell variable.
E.g:

```
ganeshell> set LAYER FSAL
```

Then, "help" will display the list of commands available for this layer. To obtains some help for a command, launch it with the "-h" option. E.g:

```
ganeshell> init_fs -h
usage: init_fs [options] <ganesha_config_file>
options :
        -h print this help
        -v verbose mode
```

Each layer need to be initialized. Most of them provide an "init" call, that takes a GANESHA configuration file as argument. E.g:

```
ganeshell> init_fs -v /etc/posix.ganesha.nfsd.conf
Filesystem initialization...
07/05/2008 13:27:39 : ganeshell-25462[shell] :Worker successfuly connected to database
Current directory is "/" (@A552000000000000CD03A7470000000000208000000000000200000000000000
```

2.1.3 Layer's debug level

The special shell variable `DEBUG_LEVEL` is specific to each layer. Setting it will impact the next calls to this layer (even if the layer is called by an upper layer).

In the following example, FSAL will be in FULL_DEBUG mode, and Cache_inode will be in EVENT level:

```
# FSAL initialization
set LAYER FSAL
set DEBUG_LEVEL NIV_FULL_DEBUG
init_fs /etc/posix.ganesha.nfsd.conf

# Metadata cache initialization
set LAYER Cache_inode
set DEBUG_LEVEL NIV_EVENT
init_cache /etc/ganesha/posix.ganesha.nfsd.conf
```

2.2 Variables

Ganeshell provides variables management, so you can create custom variables, affect values to them and use their value afterward as argument of a command, a condition, ... Variable values are stored as strings.

You can list all defined variables with the shell built-in command `varlist`.

2.2.1 Getting, setting, deleting a variable

- To get the value of a variable, place a `$` sign before its name:

  ```
  ganeshell> echo $LINE
  42
  ```

- To set the value of a variable, execute:

```
set <var_name> <expression1> [expression2] ...

Example:
ganeshell> set A 5
```

This will create the variable if it does not exist, else it will override its previous value.

Expressions can be strings, other variables, command returns...

Note that `set` accepts several expressions that will be concatenated in the target variable :

```
ganeshell> set MYVAR "A:" 1345 " L:" $LINE
ganeshell> echo $MYVAR
A:1345 L:28
```

- Finally, you can destroy a variable definition using `unset`:

```
ganeshell> unset MYVAR
ganeshell> echo $MYVAR
******* ERROR in <stdin> line 49: Undefined variable "MYVAR"
```

2.2.2 Special variables

Some special variables are defined and interpreted by shell:

- $? and $STATUS: contain the status returned by the last operation

```
E.g:
ganeshell>toto
******* ERROR in <stdin> line 3: toto: command not found
ganeshell>echo $?
-2
```

- $SHELLID: in a multi-instance (multi-thread) shell, this gives the index for the current thread (first is 0);
- $PROMPT: you can modify the prompt string with this variable. This may me useful when several threads are writing to shell standard output

```
E.g:
ganeshell> set PROMPT "thread#" $SHELLID ">"
thread#0>
```

- $INTERACTIVE: indicates whether the shell is in interactive or batch mode ;

- $INPUT: indicates the current input file for the shell (<stdin> is returned in interactive mode). Setting the value of this variable will result in executing the given file as a ganeshell script

  ```
  E.g:
  ganeshell> set INPUT /tmp/my_script.gsh
  ```

- $LINE: returns the current line number in script or standard output;
- $DEBUG_LEVEL or $DBG_LVL: debug level of the current layer;
- $VERBOSE: setting this variable will enable/disable shell verbose. This mode is similar to "`set -o xtrace`" in common UNIX shells: each executed command and its returned status are displayed on standard error output. This does not affect layer's trace level.

2.3 Expressions

2.3.1 Expression types

Several expression types can be used for affecting a value to a variable, or as a command argument:

- any unquoted sequence of characters, with no blank (space or tab)

  ```
  Example:
  ganeshell> set A 1
  ganeshell> echo $A
  1
  ganeshell> set X 2.345
  ganeshell> echo $X
  2.345
  ```

- single or double quoted strings

  ```
  Examples:
  ganeshell> set D "This is a double-quoted string"
  ganeshell> echo $D
  This is a double-quoted string

  ganeshell> set S 'This is a single-quoted string'
  ganeshell> echo $S
  This is a single-quoted string
  ```

- variable value: a $ sign followed by a variable name

  ```
  Example:
  ganeshell> set B $A
  ganeshell> echo $B
  1
  ```

- command return: a backquoted string

  ```
  Example:
  ganeshell> set DATE "current time is: " ‘shell date‘
  ganeshell> echo $DATE
  current time is: Wed May  7 14:55:49 CEST 2008
  ```

Note that you can escape a character in an expression, using a backslash:

```
ganeshell> set E "There are some \"escaped\" caracters here !"
ganeshell> echo $E
There are some "escaped" caracters here !
```

Variable references (`$<varname>`) are not interpreted when they are used in a single or double-quoted string.

```
ganeshell> set STR "The value of A is $A"
ganeshell> echo $STR
The value of A is $A
```

If you want to display the value of A, reference it outside the string:

```
ganeshell> set STR "The value of A is " $A
ganeshell> echo $STR
The value of A is 1
```

In a back-quoted string, they are however interpreted at command execution time.

2.3.2 Handling expressions

Several built-in commands can be used for handling expressions:

- `chomp`: remove newline chars at the end of an expression. It is useful for cleaning expressions returned by an external command:

  ```
  ganeshell> set RET ‘shell date‘
  ganeshell>echo "returned:[" $RET "]"
  returned:[Wed May  7 14:55:49 CEST 2008
  ]
  ganeshell> set RET ‘chomp $RET‘
  ganeshell>echo "returned:[" $RET "]"
  returned:[Wed May  7 14:55:49 CEST 2008]
  ```

- `wc`: count the number of characters and lines of an expression
- `diff`: compare two expressions
- `eq`: test if two expressions are the same (status is 0 if different, 1 if equal, -1 on error)

```
ganeshell> eq "ABCD" "ABCD"
ganeshell> echo $?
1
ganeshell> eq "ABCD" "abcd"
ganeshell> echo $?
0

# case insensitive string compare
ganeshell> eq -i "ABCD" "abcd"
ganeshell> echo $?
1

# numerical compare
ganeshell> eq -n 000001 1
ganeshell> echo $?
1
```

- `ne`: test if two expressions are different (status is 1 if different, 0 if equal, -1 on error)

2.4 Conditional execution

ganeshell provides a very basic structure for conditional execution.
It's syntax is:

```
if command0 ? command1 [: command2]
```

`command0` is first executed. If its return code is not null, then `command1` is executed, else `command2` (optional) is launched.

Basically, `command0` can be a test command like `eq` or `ne`.

```
Example:
ganeshell> cd ..
ganeshell> if ne -n $STATUS 0 ? print "cd command ERROR" : print "cd command OK"
```

2.5 Time routines

Its often useful to benchmark filesystem performances. So the shell provides calls for measuring command execution time:

- `time <command>`: measures and display the time for executing a command.

  ```
  Example:
  ganeshell> time init_fs /root/posix.ganesha.nfsd.conf
  Execution time for command "init_fs": 0.027679 s
  ```

- `timer start|stop|print`: this timer makes it possible to measure the time since it was started.

  ```
  Example:
  ganeshell> timer start
  ganeshell> cd /tmp
  ganeshell> ls
  ganeshell> timer print
  0.804578 s
  ganeshell> cd ..
  ganeshell> ls
  ganeshell> timer stop
  ganeshell> timer print
  1.423081 s
  ```

- `sleep <seconds>`: suspends shell execution for a given amount of seconds

2.6 Batch execution

For executing a ganeshell script file, simply give it as argument to the `<fsname>.ganeshell` command.

```
bash$ ./posix.ganeshell /tmp/myscript.gsh
```

You can also execute a script from the interactive mode, by setting the `INPUT` variable.

```
ganeshell> set INPUT /tmp/myscript.gsh
```

The script is then executed in batch mode. By default, the ganeshell processus terminates when the script is finished. You can however avoid this by using the '`interactive`' command: writting this command at the end of your script will result in giving back control to interactive mode, so you can execute other commands after the script run.

This can also been used for writting initialization scripts with repetitive "boot straps" operations. Here is an exemple of such a script:

```
set CONFIG_FILE /etc/ganesha/posix.ganesha.nfsd.conf
set LAYER FSAL
init_fs $CONFIG_FILE
set LAYER Cache_inode
init_cache $CONFIG_FILE
set LAYER NFS
nfs_init $CONFIG_FILE
mount /export
interactive
```

2.7 Multi-threaded batch execution

2.7.1 Command line

For starting ganeshell with multiple threads, you can launch it with as many scripts as threads wanted:

```
bash$ ./posix.ganeshell script-thr1.gsh script-other.gsh script-other.gsh
```

You can also launch multiple instances of the same script, using the `-n` option:

```
bash$ ./posix.ganeshell -n 4 script.gsh
```

2.7.2 Synchronization

The shell provides a unique synchronization call: `barrier`. When calling `barrier`, the thread is suspended until all other threads joined the barrier.

Note that shell variables are NOT shared between threads. They are local to each thread.

2.7.3 About layers initialization

Each layer must only be initialized once for all threads. Thus, it recommanded that the first script does all layers initialization and calls `barrier` afterward, so the other threads will start working only when everything is initialized properly.

3 Accessing layers

Ganeshell provides an access to main GANESHA layers, from Filesystem to NFS. Each shell interface implements native filesystem calls (like access, setattr, open, read, write,...) but also higher level features like current directory management (cd, pwd), user management (su) and some classical unix shell commands (cat).

This section gives you a short help for using each of these layers.

3.1 FSAL

This layer provides an access to the File System Abstraction Layer of GANESHA (FSAL), so you can make operations directly on the filesystem, without any cache effect due to the Cache_inode layer.

3.1.1 Initialization

For using the FSAL layer, first set the LAYER variable of the shell:

```
ganeshell> set LAYER FSAL
```

Then initialize the layer using a Ganesha configuration file:

```
ganeshell> init_fs /etc/posix.ganesha.nfsd.conf
```

3.1.2 Getting started

Once initialized, you can use most common shell commands like `cd`, `ls`, `pwd`, `stat`, `mkdir`,...
Type `help` to get FSAL command list.
To get help about a command, type `<command> -h`.

3.1.3 Paths and handles

FSAL layer of ganeshell makes it possible to address entries using their full or relative path. However, you can also address them using their FSAL handle (hexadecimal representation preceded with a '`@`' sign):

```
ganeshell> cd /
Current directory is "/" (@A552000000000000CD03A747FFFFFFFF02080000000020000000000000000000

# is equivalent to:
ganeshell> cd @A552000000000000CD03A747FFFFFFFF020800000000200000000000000000001B0000003000
```

The '`ls`' command has a special option '`-S`' for displaying objects handle.

3.1.4 Users management

If Ganeshell is launched with sufficient permissions for accessing the filesystem or the required credentials, you can take the identity of any user with the '`su`' command. This can be very useful for testing access rights of your filesystem.

You can give to '`su`' a uid or a user name:

```
ganeshell>su root
Changing user to : root ( uid = 0, gid = 0 )
altgroups = 1, 2, 3, 4, 6, 10
Done.

# is equivalent to:
ganeshell>su 0
Changing user to : root ( uid = 0, gid = 0 )
altgroups = 1, 2, 3, 4, 6, 10
Done.
```

3.2 Cache_inode

This layer provides an access to the Cache_inode layer of GANESHA, so you can interact directly with it, without going through network or NFS Protocol service routines.

3.2.1 Initialization

You must have initialized the FSAL layer before using the Cache_inode layer. To do this, report to the previous section (FSAL).

Then, set the LAYER variable of the shell:

```
ganeshell> set LAYER Cache_inode
```

After that, initialize the layer using a Ganesha configuration file:

```
ganeshell> init_cache /etc/ganesha/posix.ganesha.nfsd.conf
```

3.2.2 Getting started

Once initialized, you can use most common shell commands like `cd`, `ls`, `pwd`, `stat`, `mkdir`,... You can also make actions on data and metadata caches, like `data_cache`, `flush_cache`, garbagge collection, ...
Type `help` to get Cache_inode command list.
To get help about a command, type `<command> -h`.

3.2.3 Paths and handles

Like in the FSAL layer, filesystem entries can be addressed using their path or their FSAL handle beginning with '`@`'.

For displaying FSAL handles with `ls`, use the '`-H`' option. You can also display their addresses in metadata cache with the '`-L`' option.

3.3 NFS

This layer provides an access to the NFS layer of GANESHA, so you can execute NFS queries directly on the server, as if they were sent by a client.

3.3.1 Initialization

You must have initialized the FSAL and the Cache_inode layers before using the NFS layer. To do this, report to the previous sections (FSAL and Cache_inode).

Then, set the LAYER variable of the shell:

```
ganeshell> set LAYER NFS
```

After that, initialize the layer using a Ganesha configuration file:

```
ganeshell> nfs_init /etc/posix.ganesha.nfsd.conf
```

3.3.2 Getting started

This layer provides two ways of accessing the filesystem with NFS:

- You can use native NFS v2 and v3 calls (and mount protocol v1 and v3). In this mode, you need to be aware of manipulating NFS handles and structures ;-)

  ```
  ganeshell>mnt3_export
   {
    ex_dir = /tmp
    ex_groups =
      gr_name = 127.0.0.1
   }
  ganeshell>mnt3_mount /tmp
   mountres3 =
   {
    fhs_status = 0
    mountinfo =
    {
      fhandle3 = @A10000000000000000000000000000004F00A752000000000000CD03A74700000000000
      auth_flavor = 1
    }
   }
  ganeshell>nfs3_getattr @A10000000000000000000000000000004F00A752000000000000CD03A74700
   GETATTR3res =
   {
    status = 0 (NFS3_OK)
    fattr3 =
    {
      type = 2 (NF3DIR)
      mode = 01777
      nlink = 9
      uid = 0
      gid = 0
      size = 4096
      used = 8192
      rdev = 0.0
      fsid = 0xc1
      fileid = 0x2
      atime = 1210322525.000000000 (2008-05-09 10:42:05)
      mtime = 1210326484.000000000 (2008-05-09 11:48:04)
      ctime = 1210326484.000000000 (2008-05-09 11:48:04)
    }
   }
  ```

- You can also use simple shell commands (`cd`, `ls`, ...) wrapping MNT3/NFS3

protocol calls. All you have to do before is to make an inital "`mount <path>`", for beeing able to access an export. The export is then mounted as "/".
Note: you can get the list of exports using "`mnt3_export`".

```
ganeshell>mnt3_export
 {
  ex_dir = /tmp
  ex_groups =
    gr_name = 127.0.0.1
 }
ganeshell>mount /tmp
Current directory is "/"
Current File handle is "@A100000000000000000000000000000004F00A752000000000000CD03A7470C
ganeshell>ls -l
         2 drwxrwxrwx   9        0        0            4096    May  9 12:04 .
         2 drwxrwxrwx   9        0        0            4096    May  9 12:04 ..
        17 -rw-------   1     3733     5683             523    May  9 11:12 krb5cc_3733
        23 prw-------   1     3051     5683               0    Apr  9 11:15 gitapply.a2
    56ab41 dr-xr-xr-x   2        0        0            4096    Oct 31 2007  RPMS
        14 -rw-r--r--   1        0        0           15176    May  6 13:07 ganesha.sta
        13 srwxrwxrwx   1     2931     2931               0    May  9 12:01 .s.PGSQL.54
         d -rw-------   1     3733     5683             523    May  9 10:58 krb5cc_3733
         f -rwxr-x---   1     3733     5683        10165193    May  6 11:57 posix.ganes
        11 -rw-r--r--   1        0        0               0    Apr 23 11:16 casimir.cmd
   6170a1 drwxr-x---    3        0        0            4096    May  9 10:38 ganesha.dat
         c -rw-------   1     3733     5683             523    May  9 09:52 krb5cc_3733
        10 -rw-r--r--   1        0        0              10    May  9 12:05 invocateur.
        15 -rw-r--r--   1        0        0           33531    May  1 15:09 casimir.cmd
        12 -rw-------   1     2931     2931              26    May  9 12:01 .s.PGSQL.54
         b drwx------   2        0        0           16384    Oct 31 2007  lost+found
        16 -rwxr-x---   1     3733     5683         6967242    May  7 13:27 posix.ganes
        18 -rw-r--r--   1        0        0             148    Apr 29 13:59 casimir.cmd
    2b95c1 drwxrwxrwx   2        0        0            4096    Apr 23 11:15 .font-unix
     28141 drwxrwxrwx   2        0        0            4096    Apr 23 11:14 .ICE-unix
```

3.3.3 Paths and handles

Of course, if you are using native NFSv2/v3 calls, you will need to address entries using their NFS handle. You also need to solve the path by yourself using LOOKUP call.

If your are using shell wrappers, you can address entries with their full path or with a path relative to the current directory (managed by shell). You can also address entries using their NFSv3 handle, beginning with '`@`'.

```
ganeshell> cd /
Current directory is "/"
Current File handle is "@A100000000000000000000000000000004F00A752000000000000CD03A7470

# is equivalent to:
ganeshell> cd @A100000000000000000000000000000004F00A752000000000000CD03A74700000000000
```

For displaying NFS handles with `ls`, use the '`-H`' option.

3.4 NFS_remote

This layer makes it possible to use ganeshell as a NFS client.

3.4.1 Initialization

You don't need to initialize any other layer for using the NFS_remote layer.
First, set the LAYER variable of the shell:

```
ganeshell> set LAYER NFS_remote
```

After that, initialize the RPC connections to the remote NFS server for every protocol you need, by providing host name, protocol version and a port number (optional):

```
# initialize client for mount3 protocol
ganeshell> rpc_init localhost mount3 udp 2049

# initialize client for nfs3 protocol
ganeshell> rpc_init localhost nfs3 udp 2049
```

3.4.2 How to use it

You can now use the same functions as the NFS layer described in the previous section (both NFS native calls and high-level wrappers).

```
ganeshell>mount /tmp
Current directory is "/"
Current File handle is "@A100000000000000000000000000000004F00A752000000000000CD03A7470000000
ganeshell>cd RPMS
Current directory is "/RPMS"
Current File handle is "@A200000000000000000000000000000004F0012530000000000000C767CE4700000000
ganeshell>ls
.
..
topspin-ib-mod-rhel4-2.6.9-22.ELsmp-3.0.0-185.b.6.1.Bull.x86_64.rpm
mpich-1.2.7-p1.x86_64.rpm
RaidMan-8.25.x86_64.rpm
```