About:

This document is aimed to provide detail and structured information about NFSv4, ACLs and contains description of the test cases (implemented in Python) and my work about discovery NFSv4 and ACLs.

Creator:

A.V.Nesterovich@gmil.com

References:

Title	Reference
RFC7530 Network File System (NFS) Version 4 Protocol	https://www.rfc-editor.org/rfc/rfc7530.txt
RFC7531 Network File System (NFS) Version 4 External Data Representation Standard (XDR) Description	https://www.rfc-editor.org/rfc/rfc7531.txt
File System Extended Attributes in NFSv4 draft-ietf-nfsv4-xattrs-02	https://tools.ietf.org/id/draft-ietf-nfsv4-xattrs-02.txt
Mapping Between NFSv4 and Posix Draft ACLs	http://www.citi.umich.edu/projects/nfsv4/rfc/draft-ietf-nfsv4-acl-mapping-05.txt
acl - Access Control Lists (Linux man page) *POSIX	http://linux.die.net/man/5/acl
exports - NFS server export table (Linux man page)	http://linux.die.net/man/5/exports
exportfs - maintain table of exported NFS file systems(Linux man page)	http://linux.die.net/man/8/exportfs
rpc.mountd - NFS mount daemon (Linux man page)	http://linux.die.net/man/8/mountd
getfacl - get file access control lists (Linux man page)	http://linux.die.net/man/1/getfacl
setfacl - set file access control lists (Linux man page)	http://linux.die.net/man/1/setfacl
get NFSv4 file/directory access control lists (Linux man page)	http://linux.die.net/man/1/nfs4_getfacl
set NFSv4 file/directory access control lists (Linux man page)	http://linux.die.net/man/1/nfs4_setfacl
maximum number of ACL's available on a directory	https://access.redhat.com/solutions/68429

Chapter #1 - NFS4

Info

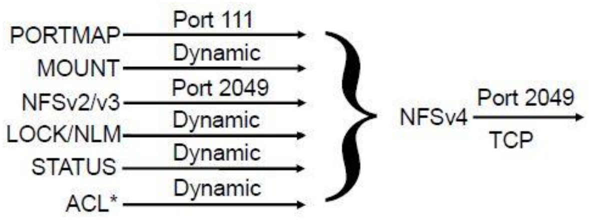
NFS is a UNIX protocol for large scale client/server file sharing. It is analogous to the server Message Block (SMB) and Common Internet File System (CIFS) protocols on Microsoft Windows. The Network File System Version 4 is a distributed filesystem protocol which owes heritage to NFSv2 and NFSv3. Unlike previous versions of NFS the present version(NFSv4) supports traditional file access while integrating support for file locking and mount protocol.

NFSv4 is the successor of NFSv3. It has been designed to work on a LAN or over the Internet.

NFS v4 provides the following benefits over NFSv3 or earlier NFS versions:

- **advanced security management** (mandates security and ACLs); Kerberos; SPKM; LIPKEY;

- **firewall friendly** (NFS v4 work by default works over TCP): the use ofportmapper dishing out arbitrary ports made it difficult for the firewall. NFSv4 changed that by consolidating most of the TCP/IP services into well-known ports which the security administrator can define in the firewall



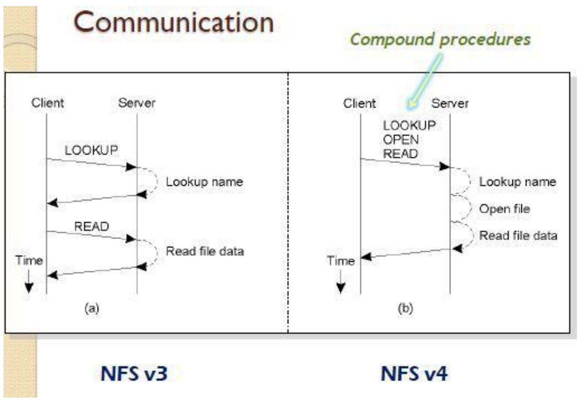
- **advanced and aggressive cache management;**

- **non Unix compatibility (Windows);**

- **easy to administer (Replication, migration);**

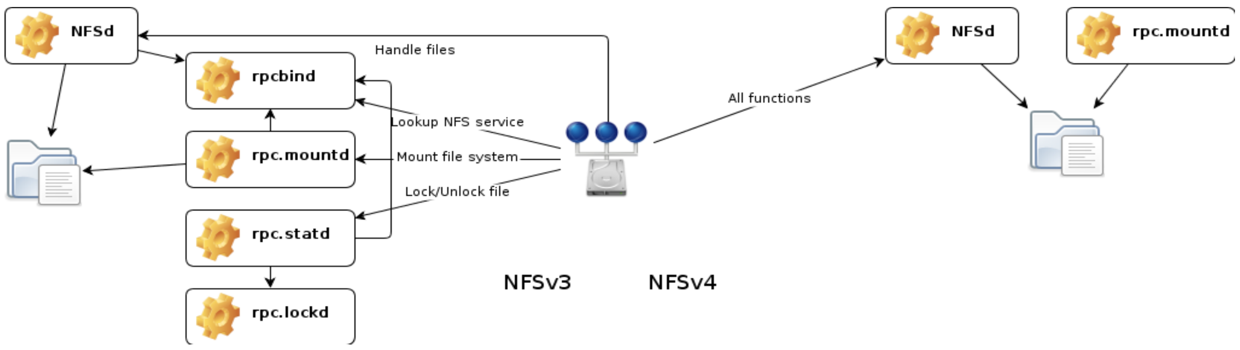
- **crash recovery (Client and server sides);**

- **performance improvements:** one key enhancement is the introduction of the COMPOUND RPC procedure which allows the NFS client to group together a bunch of file operations into a single request to the NFS server. This not only reduces the network round-trip latency, but also reduces the small little chatters of the smaller file operations.



- stateful: NFSv3 is stateless and it does not maintain the state of the NFS clients. NFSv4 is stateful and implements a mandatory locking and delegation mechanisms. Leases for locks from the servers to the clients was introduced. A lease is a time-bounded grant for the control of the state of a file and this is implemented through locks.

The NFSv3 and NFSv4 protocols are not compatible. A NFSv4 client cannot access a NFSv3 server, and vice versa. However, in order to simplify migrations from NFSv3 to NFSv4, both NFSv3 and NFSv4 services are launched by the command: **rpc.nfsd**. In the case of NFSv3 and NFSv4 clients simultaneously accessing the same server, one must be aware that two different file systems are used: there is no backward support to NFSv3 by the NFSv4 server. In order to ensure a better reliability over the Internet, NFSv4 only uses TCP. To help NFS setup for internet use, one unique network port is used on NFSv4. This predetermined port is fixed. The default is **port 2049**. With NFSv4, mount and locking services have been integrated in the NFS daemon itself.



Compare NFSv3 and NFSv4 implementations

The NFSv4 server and clinet work **without the portmap, rpc.lockd, rpc.statd** daemons. The **rpc.mountd daemon is still required** on the server.

Since NFSv4 no longer utilizes the **rpc.mountd** protocol as was used in NFSv2 and NFSv3, the mounting of file systems has changed. An NFSv4client now has the ability to see all of the exports served by the NFSv4server as a single file system, called the **NFSv4 pseudo-file system**. On Red Hat Enterprise Linux, the pseudo-file system is identified as a single, real file system, identified at export with the fsid=0 option.

A NFSv4 client communicates with corresponding NFSv4 Server via Remote Procedure Calls (RPS's). The client sends a request and gets a reply from the server. A NFSv4 server can only provide/export a single, hierarchical file system tree. If a server has to share more than one logical file system tree, the single trees are integrated in a new virtual root directory. This construction, called pseudo file system, is the one which is provided/exported to clients.

Chapter #2 - ACLs

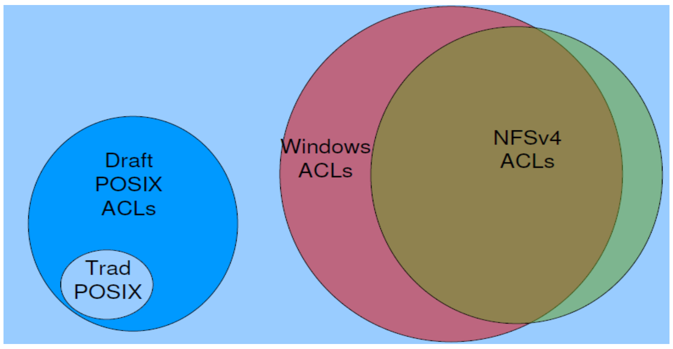
General Info

POSIX is a family of standards, specified by the IEEE, to clarify and make uniform the application programming interfaces (and ancillary issues, such as commandline shell utilities) provided by Unix-y operating systems.

ACL = Access Control List (ACLs allow a sysadmin to express nontrivial rules defining access control on objects (e.g. files or directories))

Option	**Access Model In General**	**Access Models For Filesystems**
Subject	entity which performs actions	process
Object	entity on which actions are performed	inode, i.e. file or directory or other object
Algorithm	**(subjectcredentials, objectpermissions, actionrequested)** 1) definition of format for subject credentials 2) definition of format for object permissions set of actions a subject can perform	**might use the permissions of the parent directory** 1) subject credentials = an owner, some groups 2) object permissions = an owner, an owning group,a POSIX mode and/or some kind of ACL actions e.g. Read, Write, Execute, Delete, Change Owner

Access Models For Filesystems:



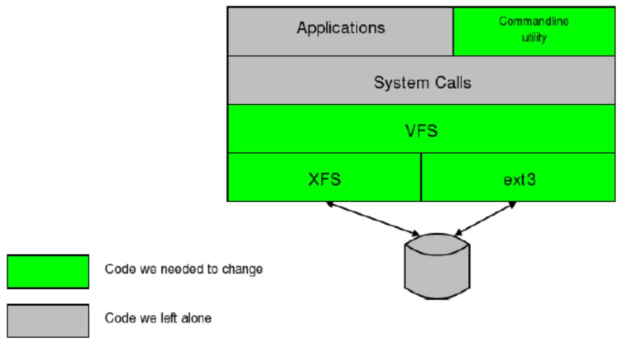
Model	**Description**
Traditional POSIX chown chmod	Owners & groups identified by UIDs, GIDs / Locally mapped on each system (e.g. /etc/passwd, NIS (Network Information System, LDAP (Lightweight Directory Access Protocol)) / 3 bit access mask / Read (r), Write (w), Execute (x) / Most actions are mapped to one of these 3 / – exceptions, e.g. Delete => Write on the parent directory / Subjects classified into one of 3 classes / − Owner = the subject's owner matches the object's, or / − Group = one of the subject's groups matches the object's owning group, or / − Other = none of the above / 1 access mask per class = 9 bits / setuid, setgid,

	sticky bits = 12 bits mode (Total) / Classify subject by matching UIDs and GIDs / - Use class to choose one of the 3 access masks / - If desired bit is set access is allowed, else denied
Draft POSIX ACLs	Draft POSIX extensions 1003.1e and 1003.2c / − never ratified / − but implemented several times − simply extends POSIX to allow more entries / Users & groups UIDs/GIDs (same) / 3 bit access mask (same) / 3 special classes (same) / − Owner (tag ACL_USER_OBJ), Group (ACL_GROUP_OBJ), Other (ACL_OTHER) / 1 entry per each special class / Any number of additional entries (ACEs) **ACE (Access Control Entries)** − General user (ACL_USER), general group (ACL_GROUP) / − Total between 3 .. _POSIX_ACL_ENTRIES_MAX (>= 16) entries / − Order not significant / Entries only allow access, never deny access / − Access not explicitly allowed is implicitly denied / If subject UID matches an ACL_USER or ACL_USER_OBJ entry, / use that / Else if subject GIDs match an ACL_GROUP_OBJ or ACL_GROUP entry, use that / Else use the ACL_OTHER entry
Windows NT ACLs **setfacl** **getfacl**	From Microsoft / − Implemented in Windows NT / − Minor changes in subsequent releases Users & groups identified by SIDs (Security Identifiers) / − Like a variable-length enormous binary UID but with global scope / − e.g. S152110043363481177238915682003330512 / 14 access mask bits / − ReadData/ListFolder, WriteData/CreateFile, AppendData/CreateFolder, /ReadExtendedAttributes, WriteExtendedAttributes, Execute/TraverseFolder, / DeleteChild, ReadAttributes, WriteAttributes, Delete, ReadPermissions, / WritePermissions, TakeOwnership, Synchronize / Variable number of **ACEs** / − Up to 64K size = ~1800 ACEs / − Entry has a SID, a type, some flags, and an access mask / − Order significant / 3 useful special classes (WellKnown SIDs) − Creator Owner, Creator Group, Everyone /Several less helpful WellKnown SIDs / − Interactive, Network, Dialup, Batch, Anonymous, Authenticated, Service etc / Any number of other SIDs − general user, general group / Entry type / − AccessAllowed = subject is allowed the access bits − AccessDenied = subject is denied the access bits / − SystemAudit = wacky audit stuff Entry flags / − INHERITED, INHERIT_ONLY, CONTAINER_INHERIT, OBJECT_INHERIT, / NO_PROPAGATE_INHERIT − supports inheritance / − SUCCESSFUL_ACCESS, FAILED_ACCESS − wacky audit stuff / ACL flags / − Actually in the containing Security Descriptor / − AUTO_INHERITED, DEFAULTED, PROTECTED / Loop through each entry / − stop if entry's SID matches any of subject's SIDs / If matching entry is Allow, access allowed / If matching entry is Deny, access denied / If no matching entry, access denied / TODO:reexpress
NFSv4 ACLs **nfs4_getfacl** **nfs4_setfacl**	Defined in NFSv4 standards / Tries to make the Windows access model usable over NFS / − without admitting it's Windows / − obviously nobody implemented it before writing the RFC / − architecture very similar to Windows, differs only in details / Users & groups identified by strings − “user@domain” or “group@domain” / − for transmission; systems expected to map these to some other local form/ like UIDs / 14 access mask bits / − Binary values identical to Windows − Names and semantics...similar...to Windows / − NFSv4.1 adds 2 more which have no equivalent in Windows / 3 useful special classes (whos) / − OWNER@, GROUP@, EVERYONE@ − Nearly identical to Windows / Several unhelpful special classes / − Blindly copied from Windows − But much less helpful in an NFS context / Variable number of other entries TODO:rephrases − General user, general group / Any number of entries / − No defined limit / Entry type

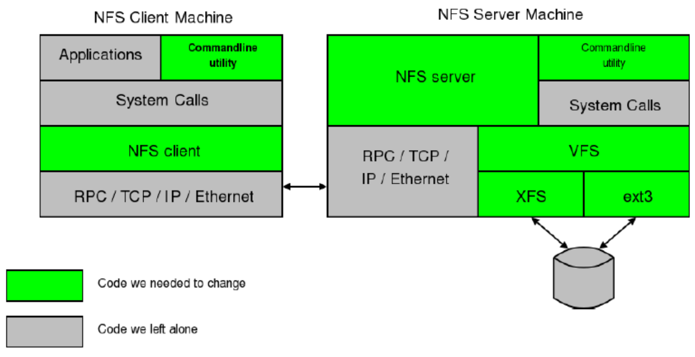
	− Blindly copied from Windows / − Including audit / Per-entry flags / − Blindly copied from Windows / − Added ACE4_IDENTIFIER_GROUP to tell apart user & group whos / Per-ACL flags − Blindly copied from Windows / − Not in NFSv4, added in 4.1

Implement NFSv4 ACLs in:

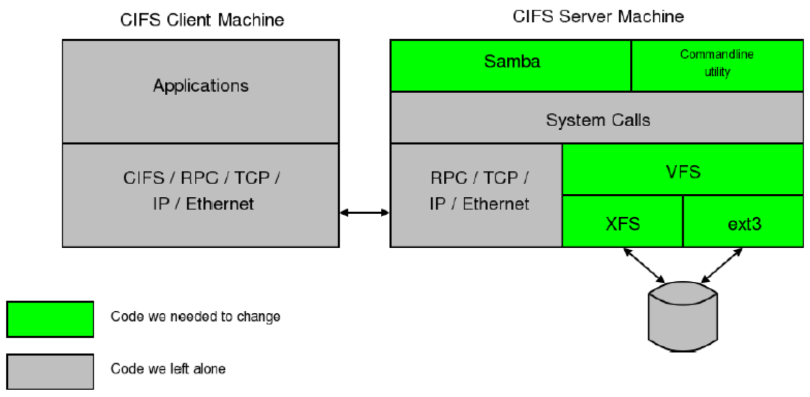
NFS4 ACL (Server)	NFS4 ACL (Client)
Server assumes underlying filesystem does POSIX ACLs − Converts to POSIX ACL when client sets an ACL − Converts from POSIX ACL when client gets an ACL − In general this conversion is lossy − Samba is doing (hopefully) the same conversion in userspace	Client presents ACLs in an unexpected manner − Nonstandard Extended Attribute, formatted as NFSv4 XDR − Need special utilities to set, print − These utilities are different to what's used on the server − Problems with cp, tar
NFSv4 ACLs are more expressive: − e.g. Deny ACEs, inheritance control / − you might learn to like that extra power / **Can a mixed Windows/Linux environment:** − with global access policies / − and files being shared between both clients / − over both NFS and CIFS protocols	



Architecture: Local Applications



Architecture: Using NFS



Architecture: Using CIFS

NFSv4 ACLs vs POSIX

ACE4_SYNCHRONIZE access bit

– obviously makes no sense on POSIX

ACE4_{READ,WRITE}_NAMED_ATTR access bits

– make no sense either, but rather less obviously!

Preserving more obscure corners of POSIX behaviour

– Sticky bit. CAP_FOWNER, CAP_CHOWN. Restricted_chown.

Doing chmod right: file_masks and the protocol

EVERYONE@ != POSIX Other class

– Far too easy to forget

Other whos (INTERACTIVE@ etc)

– Make no sense in NFS context; preserved but ignored

ACL Text Representation

List of ACEs, separated by whitespace

Each ACE is four fields separated by colons

who:accessmask:flags:type (e.g. accounts:rwax:g:allow)

1) who

2) access mask – 1char abbrevs, any order

r = read_data/list_directory

w = write_data/add_file

a = append_data/add_subdirectory

x = execute / traverse_directory,

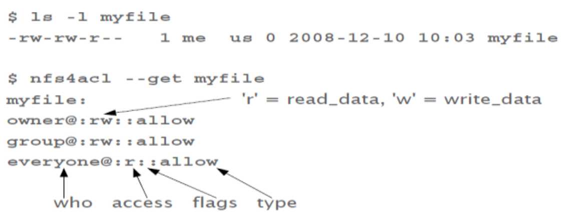
etc etc

3) Flags – 1char abbrevs, any order

g = who field names a group

4) Type

allow, deny



```
$ ls -l myfile
-rw-rw-r--    1 me   us  0  2008-12-10  10:03  myfile

$ nfs4acl --get myfile
myfile:
owner@:rw::allow
group@:rw::allow
everyone@:r::allow
```

POSIX and ACLs (Windows)

chown is for POSIX ACLs permissions (Unix)
setfacl is for ACLs (Windows) **ACEs** - NFS share permissions are governed from NFS server side

On NFS server, if file system which is exported by NSF server supports ACL and ACLs **can be read by NFS Clients,** then ACLs are utilized by client System.

For disabling ACLs on NFS share, you have to add option "no_acl" in '/etc/exportfs ' file on NFS Server. To disable it on NSF client side again use "no_acl" option during mount time.

Access ACLs: Access ACLs are used for granting permissions on any file or directory.

Default ACLs: Default ACLs are used for granting/setting access control list on a specific directory only.

***By default, if the file system being exported by an NFSv4 server supports ACLs and the NFS client can read ACLs, ACLs are utilized by the client system.

Maximum number of ACL's available on a directory on various filesystems (Supported ACL Entries) (ext2/ext3/ext4; xfs; gfs2; nfs)

The maximum number of ACL's supported on GFS2 and XFS filesystems is hard-coded at 25. This value cannot be altered without changing source code and recompiling the corresponding kernel modules.

The maximum number of ACL's on EXT2/EXT3 on RHEL3 was 32 per file/directory, but this limit was raised in RHEL4 to the maximum number that will fit in a filesystem block.

File system / Info	Restrictions
GFS2 The maximum number of ACL's on an individual file/directory on a **GFS2 filesystem**	**25** fs/gfs2/acl.h: #define GFS2_ACL_MAX_ENTRIES 25
XFS The maximum number of ACL's on an file/individual directory on an XFS filesystem	**25** fs/xfs/xfs_acl.h: #define XFS_ACL_MAX_ENTRIES 25
EXT2, EXT3, EXT4 The maximum number of ACL's on an individual file/directory on an EXT2, EXT3 or EXT4 filesystem varies with block size	**32** Based on a blocksize of 4096 (default), approximately 500 ACL's can be stored on each file/directory. *** Once the limit is reached, attempting to add additional ACL's will result in the error: setfacl: /acltest/directory: No space left on device
NFS The maximum number of ACL's supported on a file/directory exported by NFS	**1024** include/linux/nfsacl.h: #define NFS_ACL_MAX_ENTRIES 1024

The access ACL of a file system object is accessed for every access decision that involves that object. Access checking is performed on the whole path from the namespace root to the file in question. It is important that ACL access checks are efficient. To avoid frequently looking up ACL attributes and converting them from the machine-independent attribute representation to a machine-specific representation, the Ext2, Ext3, JFS, and ReiserFS implementations cache the machine-specific ACL representations. This is done in addition to the normal file system caching mechanisms, which use either the page cache, the buffer cache, or both. XFS does not use this additional layer of caching.

Most UNIX-like systems that support ACLs limit the number of ACL entries allowed to some reasonable number. ACLs with a high number of ACL entries tend to become more difficult to manage. More than a handful of ACL entries are usually an indication of bad application design. In most such cases, it makes more sense to make better use of groups instead of bloating ACLs.

The ReiserFS and JFS implementations define no limit on the number of ACL entries, so a limit is only imposed by the maximum size of EA values. The current EA size limit is 64 KiB, or 8191 ACL entries, which is too high for ACLs in practice: besides being impractical to work with, the time it would take to check access in such huge ACLs may be prohibitive.

POSIX Permissions (chown, chgrp, chmod, umask)

1) chown : change file owner and group

To check the ownership of a file or directory use ls -l

Usage: chown [-Rcfv] newowner filenames/directory. Take note only root can change the ownership.

Example:

chown linda file.txt

This will cause file.txt to now be owned by linda.

chown -R abu:sales /home/account/

This is going to make all files inside /home/account/ and its subdirectories to belong to abu and to be associated with the group sales. -R means include all subdirectories

2) chgrp : change group ownership

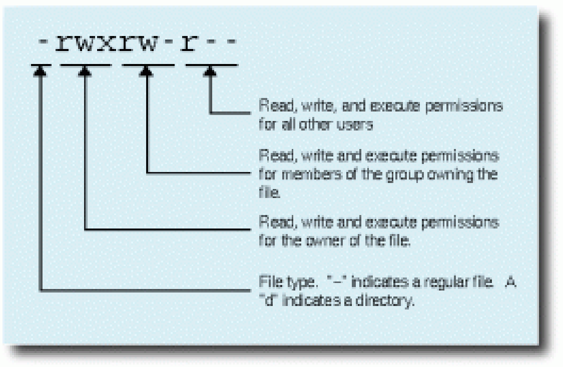
Usage : chgrp [-Rcfv] groupname foo.txt

chgrp marketing file.txt – to change the group specified to a certain document

chgrp oracle /usr/database – to change the group specified to a certain directory

chgrp -R marketing /sales/2008 – to change the group specified to a certain directory recursively

3) chmod : to change the permissions of a file or directory. Use ls -l to see the permission settings.


-rwxrw-r--
Read, write, and execute permissions for all other users
Read, write and execute permissions for members of the group owning the file.
Read, write and execute permissions for the owner of the file.
File type. "–" indicates a regular file. A "d" indicates a directory.

rwx rwx rwx = 111 111 111

rw- rw- rw- = 110 110 110

rwx --- --- = 111 000 000

and so on...

rwx = 111 in binary = 7

rw- = 110 in binary = 6

r-x = 101 in binary = 5

r-- = 100 in binary = 4

For example, if we wanted to set some_file to have read and write permission for the owner, but wanted to keep the file private from others, we would:

chmod 600 some_file

Here is a table of numbers that covers all the common settings. The ones beginning with "7" are used with programs (since they enable execution) and the rest are for other kinds **of files.**

Value	Meaning
777 (rwxrwxrwx) No restrictions on permissions. Anybody may do anything. Generally not a desirable setting.	777 (rwxrwxrwx) No restrictions on permissions. Anybody may do anything. Generally not a desirable setting.
755 (rwxr-xr-x) The file's owner may read, write, and execute the file. All others may read and execute the file. This setting is common for programs that are used by all users.	755 (rwxr-xr-x) The file's owner may read, write, and execute the file. All others may read and execute the file. This setting is common for programs that are used by all users.
700 (rwx——) The file's owner may read, write, and execute the file. Nobody else has any rights. This setting is useful for programs that only the owner may use and must be kept private from others.	700 (rwx——) The file's owner may read, write, and execute the file. Nobody else has any rights. This setting is useful for programs that only the owner may use and must be kept private from others.
666 (rw-rw-rw-) All users may read and write the file.	666 (rw-rw-rw-) All users may read and write the file.
644 (rw-r–r–) The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change.	644 (rw-r–r–) The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change.
600 (rw——-) The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private.	600 (rw——-) The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private.

Here are some useful settings **for directories:**

Value	Meaning
777 (rwxrwxrwx) No restrictions on permissions. Anybody may list files, create new files in the directory and delete files in the directory. Generally not a good setting. 755 (rwxr-xr-x) The directory owner has full access. All others may list the directory, but cannot create files nor delete them. This setting is common for directories that you wish to share with other users. 700 (rwx——) The directory owner has full access. Nobody else has any rights. This setting is useful for directories that only the owner may use and must be kept private from others.	777 (rwxrwxrwx) No restrictions on permissions. Anybody may list files, create new files in the directory and delete files in the directory. Generally not a good setting. 755 (rwxr-xr-x) The directory owner has full access. All others may list the directory, but cannot create files nor delete them. This setting is common for directories that you wish to share with other users. 700 (rwx——) The directory owner has full access. Nobody else has any rights. This setting is useful for directories that only the owner may use and must be kept private from others.

Who

The "who" is a list of letters that specifies whom you're going to be giving permissions to. These may be specified in any order.

u	The user who owns the file (this means "you.")
g	The group the file belongs to.
o	The other users
a	all of the above (an abbreviation for ugo)

u The user who owns the file (this means "you.")

g The group the file belongs to.

o The other users

a all of the above (an abbreviation for ugo)

Permissions

Of course, the permissions are the same letters that you see in the directory listing:

r	Permission to read the file.
w	Permission to write (or delete) the file.
x	Permission to execute the file, or, in the case of a directory, search it.

r Permission to read the file.

w Permission to write (or delete) the file.

x Permission to execute the file, or, in the case of a directory, search it.

Let's say we have these files:

```
-rwxrwxrwx  joe  acctg  wordmatic

-r--r--r--  joe  acctg  calcmatic
```

We'd like to remove write permission for the group and others on wordmatic, and add write and execute permission for all users on calcmatic.

Rather than try to figure out what the new permissions are and do these commands:

```
chmod go=rx wordmatic
```

```
chmod a=rwx calcmatic
```

The chmod command literally lets us add and subtract permissions from an existing set by using + or - instead of =.

Thus, we can take away the first file's write permission for the group and others with this command:

```
chmod go-w wordmatic
```

…and we can add write and execute permission to the second file for all users with:

```
chmod a+wx calcmatic
```

Copying Permissions

As one other shortcut, it's possible to tell chmod "give users of one class the same permissions that some other class has."

Let's say we have these files:

```
d------rwx  joe  acctg  sales
```

```
-rw-r--r--  joe  acctg  info.dat
```

The other users have full permissions on the sales directory. We'd like to say " the user and group should be assigned (=) the permissions belonging to others." That translates to:

```
chmod ug=o
```

Similarly, to make info.dat readable and writable to the group, we can say:

```
chmod g=u info.dat
```

(you can read this as "the group is assigned (=) the permissions currently held by the user.")

You may also use + and – to add and subtract the permissions that currently belong to a different class of user.

You can't mix the standard permissions (r, w, and x) with the coyping shortcuts. chmod will protest if you give it something like this:

```
chmod g=wu info.dat
```

4) umask = set file creation mask

The 'UMASK' is the default permission setting that is applied to your files and directories when they are created. After files and directories are created, the chmod command can be used to change the permissions to allow or disallow access as before. The UMASK is set when you login to a UNIX machine.

It is, however, possible to change your UMASK and put the UMASK in your login files so that your default permissions are always set for files when you create them.

Just like chmod, a umask works on a number. However, instead of the numbers being ADDED like chmod, with a umask the numbers are SUBTRACTED from 7.

So from chmod –

Read – 4

Write – 2

Execute – 1

If a user wants all directories to be created with rwxr-xr-x, that is

Owner == Read, Write, Execute == 7 – 4 – 2 – 1 == 0

Group == Read, Execute == 7 – 4 – 1 == 2

Others == Read, Execute == 7 – 4 – 1 == 2

Symbolic	**Numeric**	**Description**
----------	0000	no permissions
---x--x--x	0111	execute
--w--w--w-	0222	write
--wx-wx-wx	0333	write & execute
-r--r--r--	0444	read
-r-xr-xr-x	0555	read & execute
-rw-rw-rw-	0666	read & write
-rwxrwxrwx	0777	read, write, & execute
-rwxr-----	0740	user can read, write, & execute; group can only read; others have no permissions

Chapter #3 - Test cases (task)

Intro

Task verifies several candidate abilities including:

- ability to study independently by learning POSIX file systems standard

- perform data analysis by selecting important information

- design test cases and prepare documentation

- implement code based on design

Test Task

Design a set of test cases for owner / permission / content modification testing of NFSv4file system. Implement designed test cases as testing application (test suite). E.g, all tests are stored in “tests” folder and there is “main” file which run all tests and produces an output.

The results of the task are:

1. Test documentation. Use the following format:

- Name of test case

- Description

- Steps

- Expected result of each step

2. Source code of test suite

3. Logs of the latest successful tests execution

Test case example:

- Test name: Change file attributes to disable run, enable it, disable again.

- Description: test verifies that after several disable, enable actions permissions set to last value.

Acceptance criteria:

1. At least 6 test cases have to be created

2. At least 2 test cases for ACL management verification (optional)

3. Test documentation

4. Use one of the following scripting language:

a. Python (preferable)   b. Ruby   c. Perl (in OOP style)

5. Test Suite has to prepare and clean environment

6. Keep logs in log file, Short summary should be printed at the end of testing. E.g.:

TC001: Passed   TC002: Failed   TC003: Passed

7. Should be executable at any Linux-Like system

8. Please use comments in the code

Chapter #4 - Tests cases (Implementation)

Goal:

To develop the test suite and create documentation (subject NFSv4 with ACL support test automatization for Linux-like systems [server-client sides])

Test environment

Hostname	IP	Software	Description
fedora	192.168.100.182 (LAN)	Fedora 23 Workstation x86-x64 (ext4) + tools	Python 2.7, IDE Pycharm (create tests) Client NFSv4 (run tests)
rhel	192.168.100.176 (LAN)	Red Hat Enterprise Linux Server release 7.2 (Maipo) (RHEL) x86-x64 (LVM, XFS) + tools	Server NFSv4 (run tests)

Daemons NFS

	both sides	server side
user daemons	**rpc.idmapd** - This process provides NFSv4 client and server upcalls which map between on-the-wire NFSv4 names (which are strings in the form of user@domain) and local UIDs and GIDs. For idmapd to function with NFSv4, the /etc/idmapd.conf must be configured. This service is required for use with NFSv4.	**rpc.nfsd** - Allows explicit NFS versions and protocols the server advertises to be defined. It works with the Linux kernel to meet the dynamic demands of NFS clients, such as providing server threads each time an NFS client connects. This process corresponds to the nfs service. **rpc.mountd** - daemon is still needed to handle the exports, but is not involved with network communication anymore (in other words, the client connects directly with the NFS daemon).
kernel parts	NFSv4, RPC, XDR, TCP, IPv4	-

Packets and tools NFS

Name	Description	Addition
nfs-utils	The nfs-utils package provides a daemon for the kernel NFS server and related tools, which provides a much higher level of performance than the traditional Linux NFS server used by most users.	-
nfs4-acl-tools	The nfs4-acl-tools packages provide utilities for managing NFSv4 Access Control Lists (ACLs) on files and directories mounted on ACL-enabled NFSv4 file systems. These updated packages fix the following bug. This package contains commandline and GUI ACL utilities for the Linux NFSv4 client.	-
libnfsidmap	Is a library holding mulitiple methods of mapping names to id's and visa versa, mainly for NFSv4.	-
showmount	show mount information for an NFS server	-

Configs NFS

File	Description	Comment
Server side		
/etc/exports	It is a main configuration file, controls which file systems are exported to remote hosts and specifies options. This file contains a list of entries; each entry indicates a volume that is shared and how it is shared.	There must be at least one entry with fsid=0. (this will be pseudo file system's /). **acl** option use.
/etc/sysconfig/nfs	This file is used to control which ports the required RPC services run on. Here the number of kernel threads, NFSv4 support and GSS security (kerberos) for NFS can be configured.	Not used on the project
/etc/idmapd.conf	It translates user and group ids into names, and to translate user and group names. Use to modify the default "Domain" to contain DNS domain name.	Not used on the project
Client side		
/etc/fstab	This file is used to control what file systems including NFS directories are mounted when the system boots.	**acl** option use
/etc/idmapd.conf	It translates user and group ids into names, and to translate user and group names. Use to modify the default "Domain" to contain DNS domain name.	Not used on the project

Useful commands NFS

Description (Action)	Client side (Comand)	Server side (Comand)
Umount all nfs mounts on client	umount -a -t nfs	-
Check all nfs mounts on client	mount | grep nfs	-
Reexport of shares files Display of shares files	-	exportfs -r exportfs -v
Check all registered RPC programs (nfs, portmapper, mountd, ...)	rpcinfo -p	rpcinfo -p
Check mount information on NFS server	showmount -e <server IP>	showmount -e <server IP>
Display statistics kept about NFS client and server activity	nfsstat	nfsstat
Get file access control lists. For each file, getfacl displays the file name, owner, the group, and the Access Control List (ACL). If a directory has a default ACL, getfacl also displays the default ACL. Non-directories cannot have default ACLs.	getfacl -R <dir> getfacl <file>	getfacl -R <dir> getfacl <file>
This utility sets Access Control Lists (ACLs)	-	setfacl -R <options>

of files and directories.		
Display the NFSv4 Access Control List (ACL) for file (a file or directory), provided file is on a mounted NFSv4 filesystem which supports ACLs.	nfs4_getfacl <options>	nfs4_getfacl <options>
Manipulates the NFSv4 Access Control List (ACL) of one or more files (or directories), provided they are on a mounted NFSv4 filesystem which supports ACLs.	-	nfs4_setfacl <options>

Install and settings of NFSv4 (*** without Kerberos)

Server-side (hostname: rhel)

1) Install the addition tools

The libraries and header files needed for Python developmen:

```
yum install python-devel
```

psutil - is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network) in Python:

```
pip install psutil
```

2) Modify hosts file in order to resolve IP from hostname

```
vim /etc/hosts
192.168.100.176 rhel
192.168.100.182 fedora
```

```
ping fedora
PING fedora (192.168.100.182) 56(84) bytes of data.
64 bytes from fedora (192.168.100.182): icmp_seq=1 ttl=64 time=0.335 ms
```

3) In order to use ACLs enable mount option

```
vim /etc/fstab
dev/mapper/rhel_nfs-root /                    xfs    defaults,acl       0 0
mount -a
```

4) Check which loadable kernel modules NFS currently loaded

```
lsmod | grep nfs
nfs             251815  0
fscache           64987  1 nfs
nfsd            302351  1
auth_rpcgss          59314  1 nfsd
nfs_acl           12837  1 nfsd
lockd             93572  2 nfs,nfsd
grace             13288  2 nfsd,lockd
sunrpc            300421  8 nfs,nfsd,auth_rpcgss,lockd,nfs_acl
```

5) Check and show information about NFS modules:

```
modinfo nfs
filename:      /lib/modules/3.10.0-327.13.1.el7.x86_64/kernel/fs/nfs/nfs.ko
```

```
modinfo nfsv3
```

```
filename:       /lib/modules/3.10.0-327.13.1.el7.x86_64/kernel/fs/nfs/nfsv3.ko
```

```
modinfo nfsv4
```

```
filename:       /lib/modules/3.10.0-327.13.1.el7.x86_64/kernel/fs/nfs/nfsv4.ko
```

```
modinfo nfsd
```

```
filename:       /lib/modules/3.10.0-327.13.1.el7.x86_64/kernel/fs/nfsd/nfsd.ko
```

```
modinfo nfs_acl
```

```
filename:       /lib/modules/3.10.0-327.13.1.el7.x86_64/kernel/fs/nfs_common/nfs_acl.ko
```

```
modinfo nfs_layout_flexfiles
```

```
filename:       /lib/modules/3.10.0-327.13.1.el7.x86_64/kernel/fs/nfs/flexfilelayout/nfs_layout_flexfiles.ko
```

```
modinfo nfs_layout_nfsv41_files
```

```
filename:       /lib/modules/3.10.0-327.13.1.el7.x86_64/kernel/fs/nfs/filelayout/nfs_layout_nfsv41_files.ko
```

6) Check linux kernel for ACL support (find **=Y* option in accordance with the task)

```
uname -a
```

```
Linux rhel 3.10.0-327.13.1.el7.x86_64 #1 SMP Mon Feb 29 13:22:02 EST 2016 x86_64 x86_64 x86_64 GNU/Linux
```

```
grep -i acl /boot/config*
```

```
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_EXT4_FS_POSIX_ACL=y
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_XFS_POSIX_ACL=y
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_BTRFS_FS_POSIX_ACL=y
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_FS_POSIX_ACL=y
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_GENERIC_ACL=y
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_TMPFS_POSIX_ACL=y
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_NFS_V3_ACL=y
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_NFSD_V2_ACL=y
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_NFSD_V3_ACL=y
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_NFS_ACL_SUPPORT=m
/boot/config-3.10.0-327.13.1.el7.x86_64:CONFIG_CIFS_ACL=y
```

If there is N instead of Y, then it means linux kernel doesn’t support ACL and need to be recompiled in accordance with the task.

7) Install NFS server and tools

```
yum install nfs-utils nfs4-acl-tools libnfsidmap
```

8) Check and enable NFS server services

```
systemctl list-unit-files | grep nfs
```

```
proc-fs-nfsd.mount                   static
var-lib-nfs-rpc_pipefs.mount         static
```

```
nfs-blkmap.service            disabled
nfs-config.service            static
nfs-idmap.service             static
nfs-idmapd.service            static
nfs-lock.service              static
nfs-mountd.service            static
nfs-rquotad.service           disabled
nfs-secure-server.service     static
nfs-secure.service            static
nfs-server.service            disabled
nfs-utils.service             static
nfs.service                   disabled
nfslock.service               static
nfs-client.target             enabled
```

```
systemctl enable nfs-server.service
systemctl enable nfs.service
systemctl start nfs-server.service
systemctl start nfs.service
```

9) Create NFS share and change permissions (the export filesystem) - a directories to share with client servers

```
mkdir -p /nfs
chmod a+rwxt /export
```

10) Share directories of NFS server for any (LAN, WAN, ...). Exports - NFS server export table.

```
vim /etc/exports
/nfs *(rw,fsid=0,nohide, no_root_squash, insecure,no_subtree_check,sync)
```

/nfs

***** - users from any IP address of client machine are allowed to mount directories (means any client)

rw - allow both read and write requests on this NFS volume. The default is to disallow any request which changes the filesystem.

fsid=0 - export a directory over NFSv4. NFSv4 has a concept of a root of the overall exported filesystem. The export point exported with fsid=0 will be used as this root. The /nfs directory will be root for clients. For example, if you got /nfs/tests subdirectory, then client would see them as /tests directory. NFS needs to be able to identify each filesystem that it exports. For NFSv4, there is a distinguished filesystem which is the root of all exported filesystem. This is specified with fsid=root or fsid=0 both of which mean exactly the same thing. Only for "root" directory.

nohide - setting the nohide option on a filesystem causes it not to be hidden, and an appropriately authorised client will be able to move from the parent to that filesystem without noticing the change.

no_root_squash - turn off root squashing. This option is mainly useful for diskless clients. By default, any file request made by user root on the client machine is treated as by user nobody on the server. (Exactly which UID the request is mapped to depends on the UID of user "nobody" on the server, not the client.) If no_root_squash is selected, then root on the client machine will have the same level of access to the files on the system as root on the server.

insecure - option in this entry allows clients with NFS implementations that don't use a reserved port for NFS.

no_subtree_check - this option disables subtree checking, which has mild security implications, but can improve reliability in some circumstances. If a subdirectory of a filesystem is exported, but the whole filesystem isn't then whenever a NFS request arrives, the server must check not only that the accessed file is in the appropriate filesystem (which is easy) but also that it is in the exported tree (which is harder). This check is called the subtree_check.

sync - reply to requests only after the changes have been committed to stable storage (if async - improve performance, but at the cost that an unclean server restart (i.e. a crash) can cause data to be lost or corrupted). All changes to the according filesystem are immediately flushed to disk; the respective write operations are being waited for.

11) Restart NFS server services

```
systemctl restart nfs-server.service
systemctl restart nfs.service
```

12) Reexport all directories after modifying /etc/exports and display a list of shares files and export options on a NFS server

```
exportfs -r
exportfs -v
```

```
/nfs<world>(rw,wdelay,nohide,insecure,no_root_squash,no_subtree_check,fsid=0,sec=sys,rw,insecure,no_root_squash,no_all_squash)
/nfs/tests<world>(rw,wdelay,nohide,insecure,root_squash,no_subtree_check,fsid=0,sec=sys,rw,insecure,root_squash,no_all_squash)
```

13) Config or Disable firewall (firewalld or iptables services) on NFS server to allow client servers to access NFS shares. Open TCP port # 2049 which is used by NFSv4.

a) Config firewall

```
firewall-cmd --permanent --add-service nfs      *** need only one for remote mount via TCP port 2049
firewall-cmd --permanent --add-service rpc-bind
firewall-cmd --permanent --add-service mountd
firewall-cmd --reload
firewall-cmd --list-all
```

```
public (default, active)
interfaces: ens192
```

```
sources:
services: dhcpv6-client mountd nfs rpc-bind ssh
ports:
masquerade: no
forward-ports:
icmp-blocks:
rich rules:
```

```
cat /etc/services | grep mountd
```

```
mountd        20048/tcp          # NFS mount protocol
mountd        20048/udp          # NFS mount protocol
```

```
cat /etc/services | grep nfs
```

```
nfs           2049/tcp     nfsd shilp     # Network File System
nfs           2049/udp     nfsd shilp     # Network File System
nfs           2049/sctp    nfsd shilp     # Network File System
```

```
cat /etc/services | grep rpcbind
```

```
sunrpc        111/tcp      portmapper rpcbind     # RPC 4.0 portmapper TCP
sunrpc        111/udp      portmapper rpcbind     # RPC 4.0 portmapper UDP
```

b) Disable firewall

```
systemctl disable firewalld
systemctl stop firewalld
systemctl status firewalld
```

```
systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
  Active: inactive (dead)
```

14) Disable and SELinux

```
vim /etc/selinux/config
```

```
SELINUX=disable
***
```

```
sestatus
```

```
SELinux status:               disabled
```

15) Check all registered RPC programs (nfs, portmapper, mountd)

```
rpcinfo -p

  program vers proto   port  service

   100000    4   tcp    111  portmapper

   100000    4   udp    111  portmapper

   100024    1   udp  33569  status

   100024    1   tcp  41527  status

   100005    1   udp  20048  mountd

   100005    1   tcp  20048  mountd

   100003    4   tcp   2049  nfs

   100003    4   udp   2049  nfs

   100227    3   tcp   2049  nfs_acl

    100227    3   udp   2049  nfs_acl

   100021    1   udp  39271  nlockmgr

   100021    1   tcp  60402  nlockmgr
```

16) Set UNIX extended ACLs (add r/w permissions for user: he on directory: /nfs)

***By default, if the file system being exported by an NFSv4 server supports ACLs and the NFS client can read ACLs, ACLs are utilized by the client system.

```
setfacl -R -m u:he:rwx /nfs
```

17) Check UNIX extended ACLs

```
getfacl -R /nfs
```

***By default, if the file system being exported by an NFSv4 server supports ACLs and the NFS client can read ACLs, ACLs are utilized by the client system.

```
getfacl: Removing leading '/' from absolute path names

 # file: nfs

# owner: root

# group: root

user::rwx

user:he:rwx

group::r-x
```

```
mask::rwx
other::r-x
# file: nfs/tests
# owner: root
# group: root
user::rwx
user:he:rwx
group::r-x
mask::rwx
other::r-x
# file: nfs/tests/music_for_programming_00-manifesto.mp3
# owner: he
# group: he
user::rwx
user:he:rwx
group::rwx
mask::rwx
other::rwx
# file: nfs/DDNTestTaskE1-E2-true.pdf
# owner: he
# group: he
user::rwx
user:he:rwx
group::rwx
mask::rwx
other::r-x
```

18) Install and settings addition software need for run testcases

a) rsh - remote shell access (need in order to receive commands from remote client)

Install rsh and rshd:

```
yum install rsh rsh-server
rpm -qa | grep rsh
rsh-server-0.17-76.el7_1.1.x86_64
rsh-0.17-76.el7_1.1.x86_64
```

Start rsh-server daemons:

```
systemctl enable rsh.socket
systemctl enable rlogin.socket
systemctl enable rexec.socket
systemctl start rsh.socket
systemctl start rlogin.socket
systemctl start rexec.socket
systemctl status rsh.socket
● rsh.socket - Remote Shell Facilities Activation Socket
  Loaded: loaded (/usr/lib/systemd/system/rsh.socket; enabled; vendor preset: disabled)
  Active: active (listening)
systemctl status rlogin.socket
● rlogin.socket - Remote Login Facilities Activation Socket
  Loaded: loaded (/usr/lib/systemd/system/rlogin.socket; enabled; vendor preset: disabled)
  Active: active (listening)
systemctl status rexec.socket
● rexec.socket - Remote Execution Facilities Activation Socket
  Loaded: loaded (/usr/lib/systemd/system/rexec.socket; enabled; vendor preset: disabled)
  Active: active (listening)
```

Configure rsh-server:

vim /root/.rhosts - allow the user root on the client fedora to log in as root on the target (server)

```
fedora root
```

vim /etc/securetty - enable external root user to execute the command (lists terminals from which root can log in)

```
rsh
rexec
rlogin
```

Client-side (hostname: fedora)

1) Install the addition tools

The libraries and header files needed for Python developmen:

```
yum install python-devel
```

psutil - is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network) in Python:

```
pip install psutil
```

2) Modify hosts file in order to resolve IP from hostname

```
vim /etc/hosts
```

```
192.168.100.176 rhel
192.168.100.182 fedora
```

```
ping rhel
```

```
PING rhel (192.168.100.176) 56(84) bytes of data.
64 bytes from rhel (192.168.100.176): icmp_seq=1 ttl=64 time=0.594 ms
***
```

3) Disable firewall

```
systemctl disable firewalld
systemctl stop firewalld
systemctl status firewalld
```

```
systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
  Active: inactive (dead)
```

4) Disable and SELinux

```
vim /etc/selinux/config
```

```
SELINUX=disable
***
```

```
sestatus
```

```
SELinux status:              disabled
```

5) Install nfs utils for client NFS service

```
yum install nfs-utils nfs4-acl-tools libnfsidmap
```

6) **Enable and start NFS service**

```
systemctl enable nfs-client.target

systemctl start nfs-client.target
```

7) **Check nfs shares to the clients on NFS server**

```
showmount -e 192.168.100.176
```

```
Export list for 192.168.100.176:

/nfs/tests *

/nfs      *
```

8) **Mount the exported file system**

```
mount -t nfs4 192.168.100.176:/ /nfs
```

Observe that only "/" is given instead of the actual exported path name

9) **Check mounted NFSv4 file system**

```
mount | grep nfs
```

```
nfsd on /proc/fs/nfsd type nfsd (rw,relatime)

sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)

192.168.100.176:/ on /nfs type nfs4
(rw,relatime,vers=4.1,rsize=524288,wsize=524288,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=192.168.100.182,local_lock=none,addr=192.168.100.176) *** ver 4.1
```

```
df -hT | grep nfs
```

```
192.168.100.176:/   nfs4      37G  5.5G   32G  15% /nfs
```

10) **Mount NFS file system permanentrly in order to mount after reboot system**

```
vim /etc/fstab
```

```
192.168.100.176:/   /nfs   nfs4 _netdev,auto  0  0
```

```
mount -a
```

11) **Check all registered RPC programs**

```
sdrpcinfo -p
```

```
 program vers proto   port  service

    100000    4   tcp    111  portmapper

    100000    4   udp    111  portmapper
```

12) **Install and settings addition software need for run testcases**

a) rsh - remote shell access (need in order to execute commands on remote server)

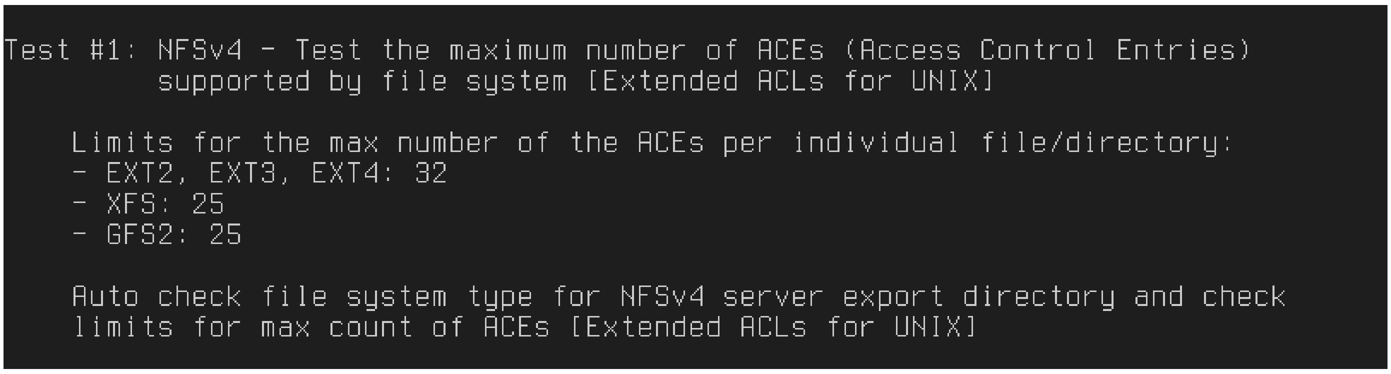
Install rsh and rshd:

```
yum install rsh rsh-server

rpm -qa | grep rsh

rsh-server-0.17-76.el7_1.1.x86_64

rsh-0.17-76.el7_1.1.x86_64
```

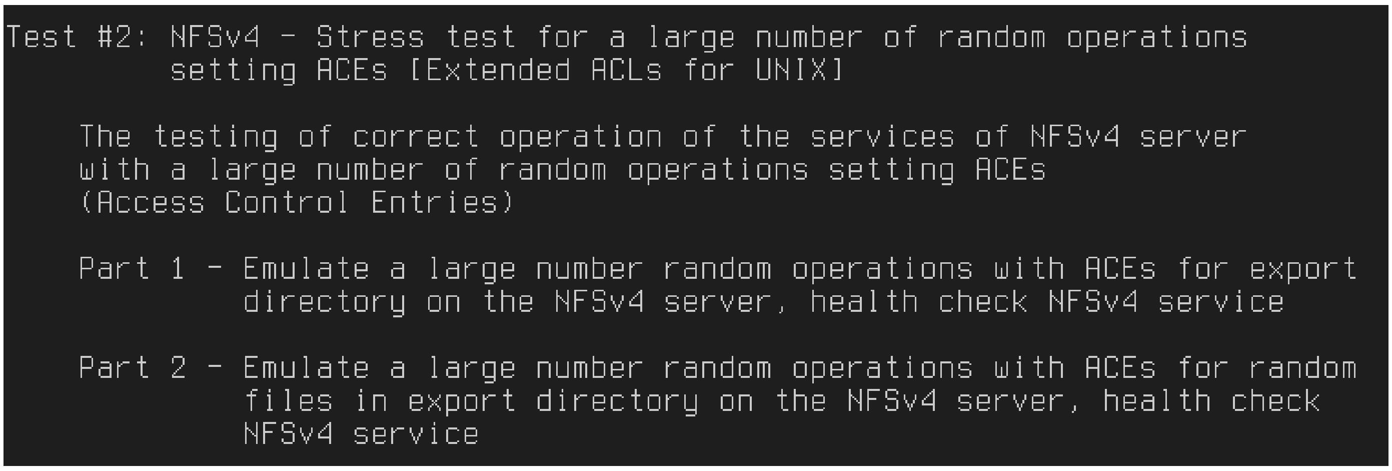
13) Check UNIX extended ACLs

```
getfacl -R /nfs
```

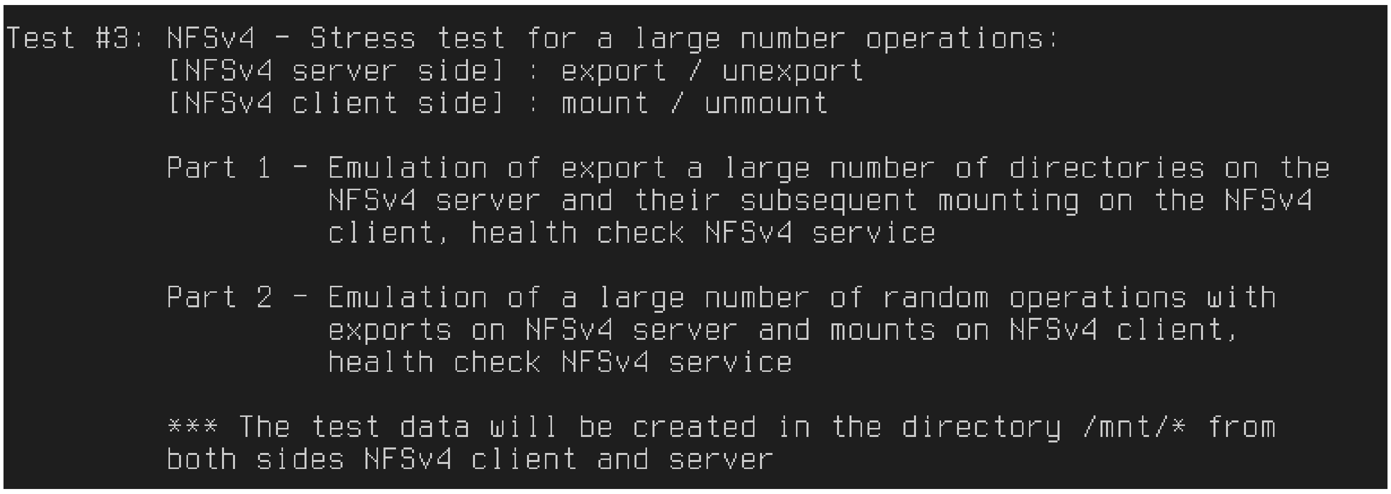
***By default, if the file system being exported by an NFSv4 server supports UNIX extended ACLs and the NFS client can read ACLs, ACLs are utilized by the client system.



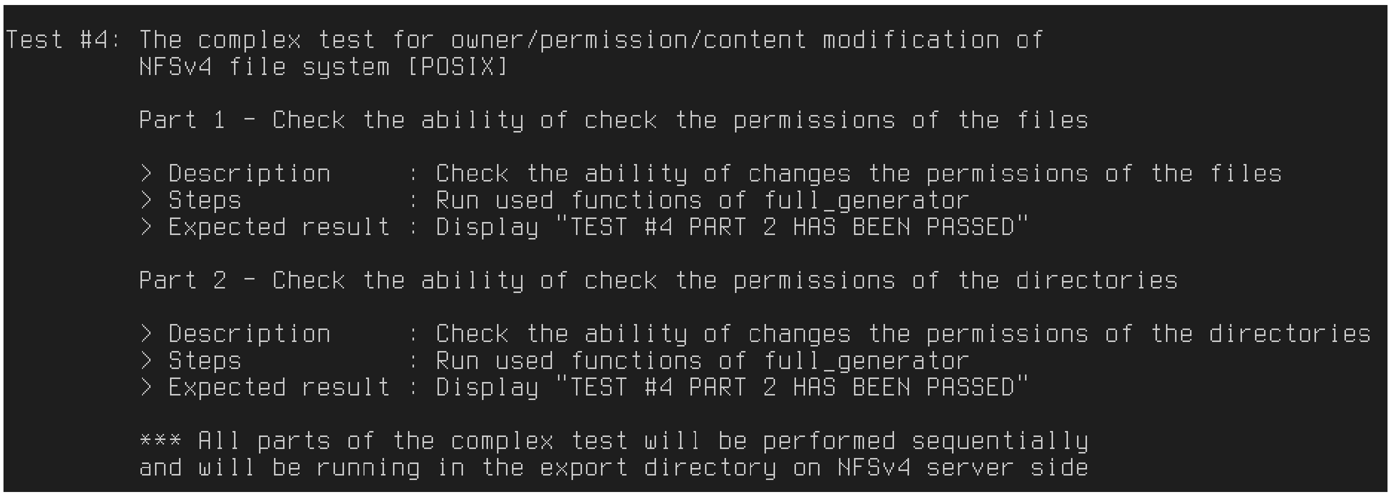
```
Test #1: NFSv4 - Test the maximum number of ACEs (Access Control Entries)
         supported by file system [Extended ACLs for UNIX]

    Limits for the max number of the ACEs per individual file/directory:
    - EXT2, EXT3, EXT4: 32
    - XFS: 25
    - GFS2: 25

    Auto check file system type for NFSv4 server export directory and check
    limits for max count of ACEs [Extended ACLs for UNIX]
```



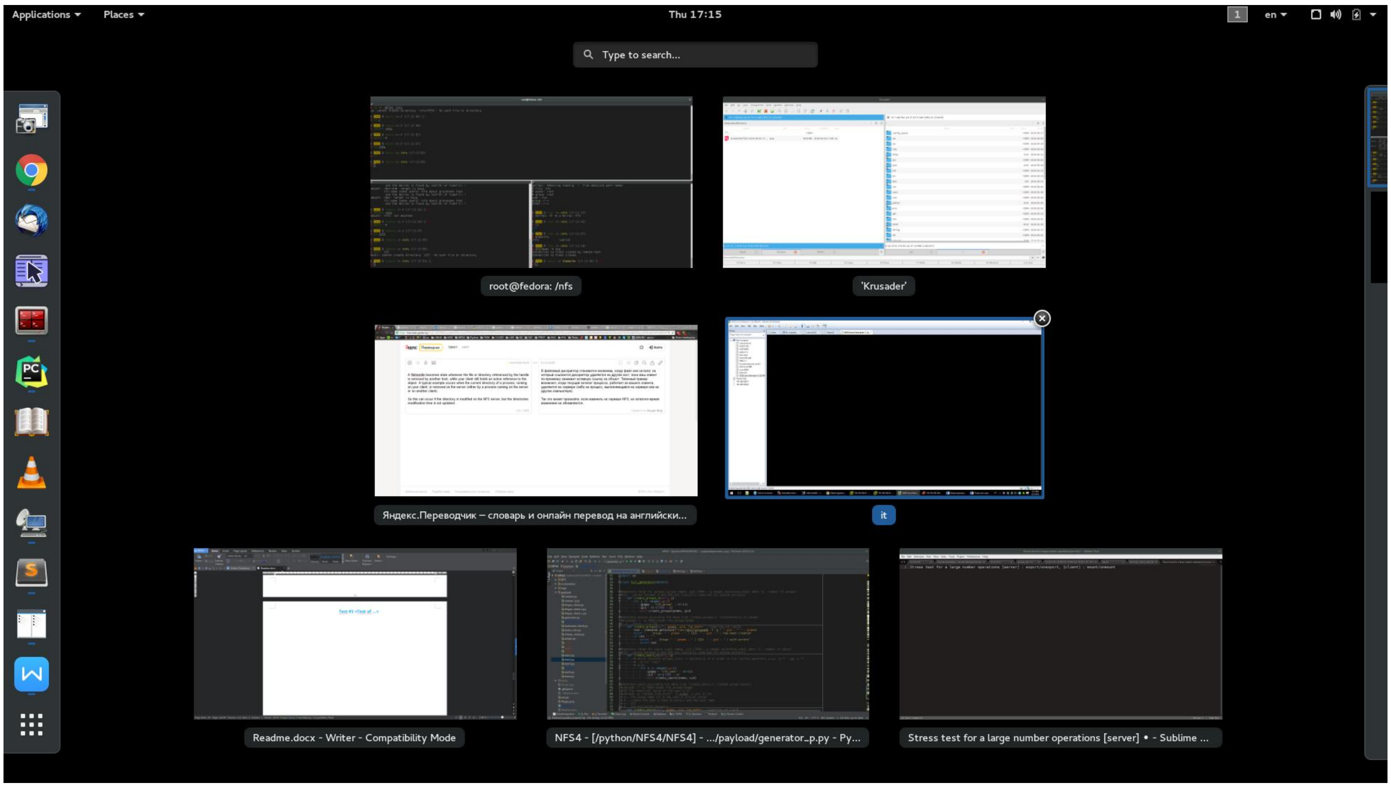
```
Test #2: NFSv4 - Stress test for a large number of random operations
         setting ACEs [Extended ACLs for UNIX]

    The testing of correct operation of the services of NFSv4 server
    with a large number of random operations setting ACEs
    (Access Control Entries)

    Part 1 - Emulate a large number random operations with ACEs for export
             directory on the NFSv4 server, health check NFSv4 service

    Part 2 - Emulate a large number random operations with ACEs for random
             files in export directory on the NFSv4 server, health check
             NFSv4 service
```

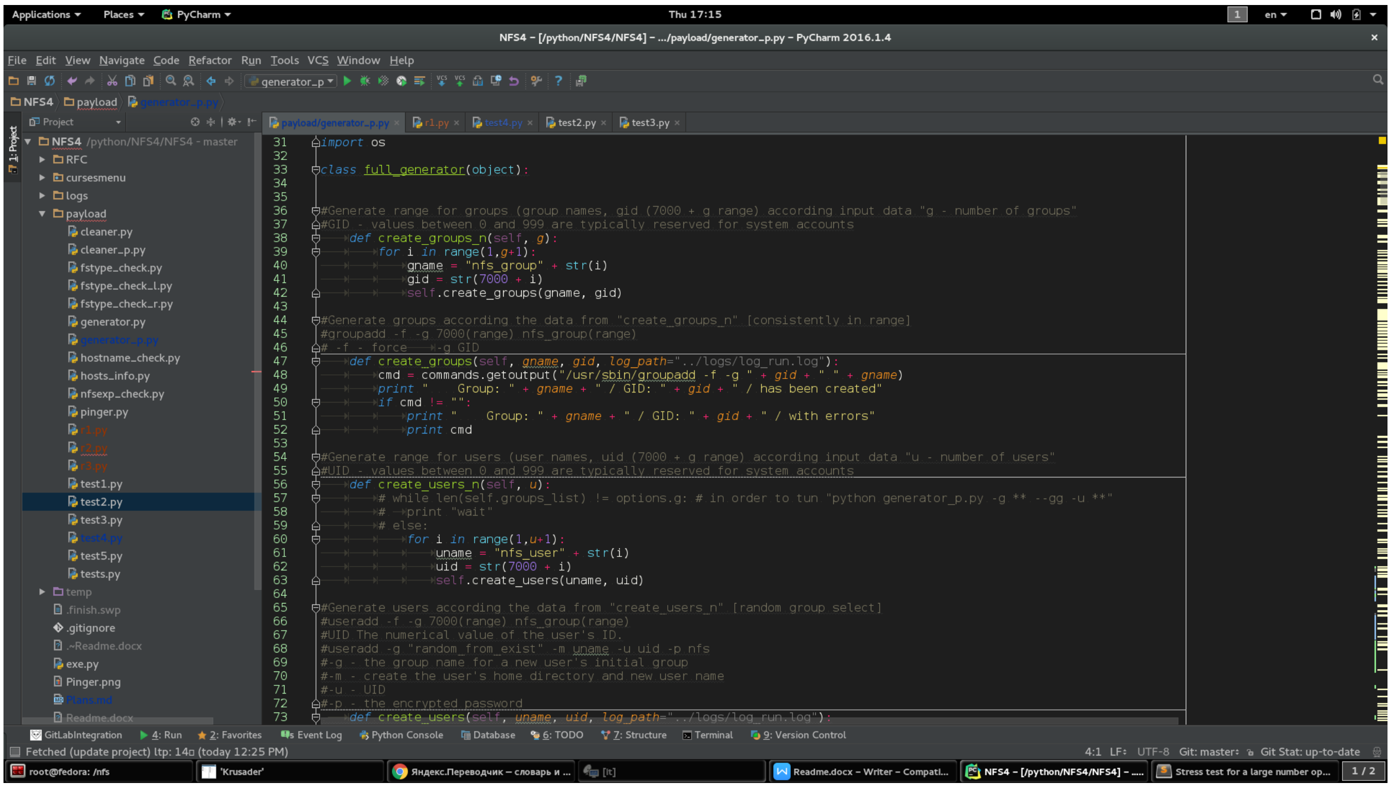


```
Test #3: NFSv4 - Stress test for a large number operations:
         [NFSv4 server side] : export / unexport
         [NFSv4 client side] : mount / unmount

         Part 1 - Emulation of export a large number of directories on the
                  NFSv4 server and their subsequent mounting on the NFSv4
                  client, health check NFSv4 service

         Part 2 - Emulation of a large number of random operations with
                  exports on NFSv4 server and mounts on NFSv4 client,
                  health check NFSv4 service

         *** The test data will be created in the directory /mnt/* from
         both sides NFSv4 client and server
```

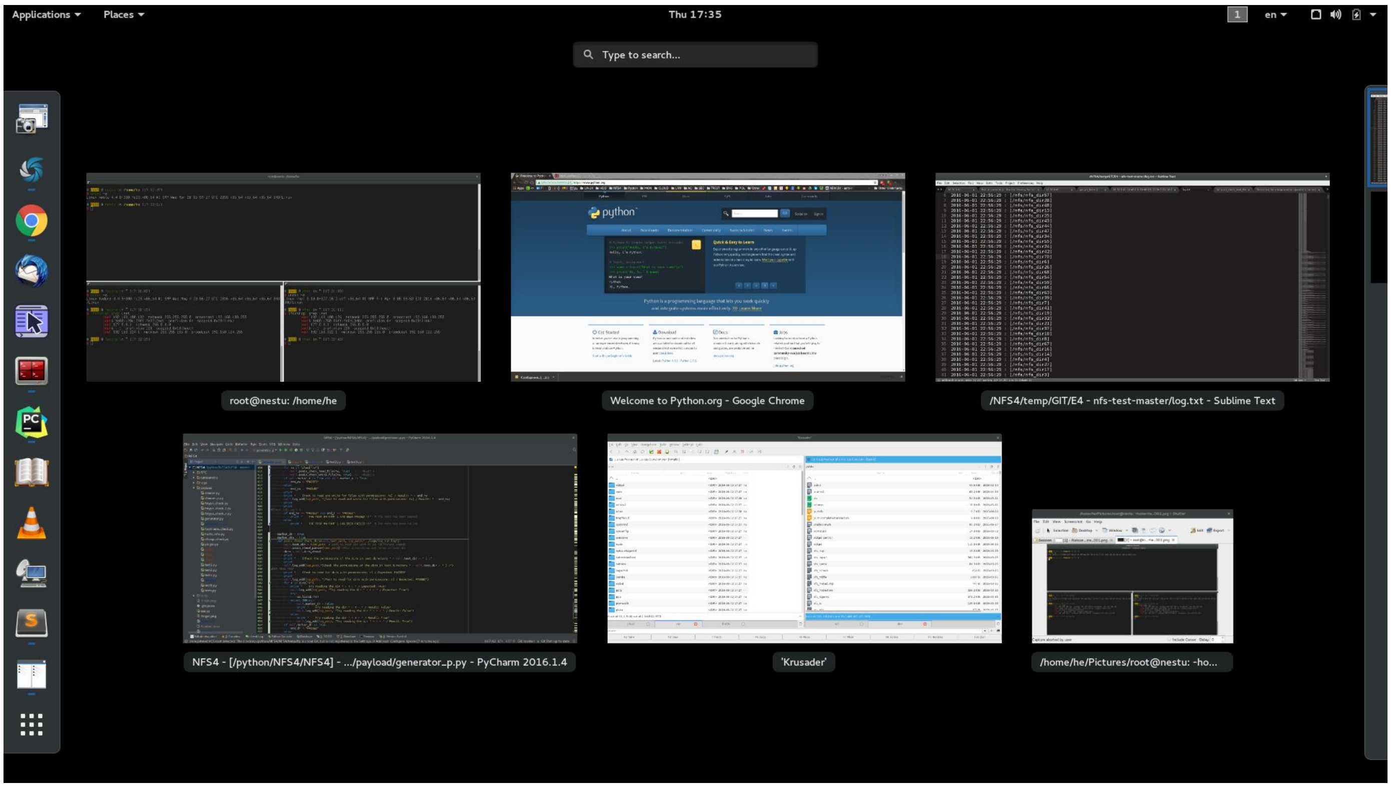


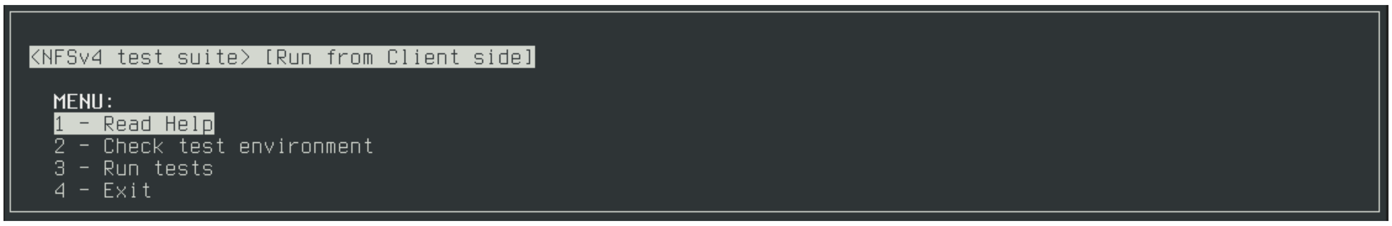
```
Test #4: The complex test for owner/permission/content modification of
         NFSv4 file system [POSIX]

         Part 1 - Check the ability of check the permissions of the files

         > Description     : Check the ability of changes the permissions of the files
         > Steps           : Run used functions of full_generator
         > Expected result : Display "TEST #4 PART 2 HAS BEEN PASSED"

         Part 2 - Check the ability of check the permissions of the directories

         > Description     : Check the ability of changes the permissions of the directories
         > Steps           : Run used functions of full_generator
         > Expected result : Display "TEST #4 PART 2 HAS BEEN PASSED"

         *** All parts of the complex test will be performed sequentially
         and will be running in the export directory on NFSv4 server side
```

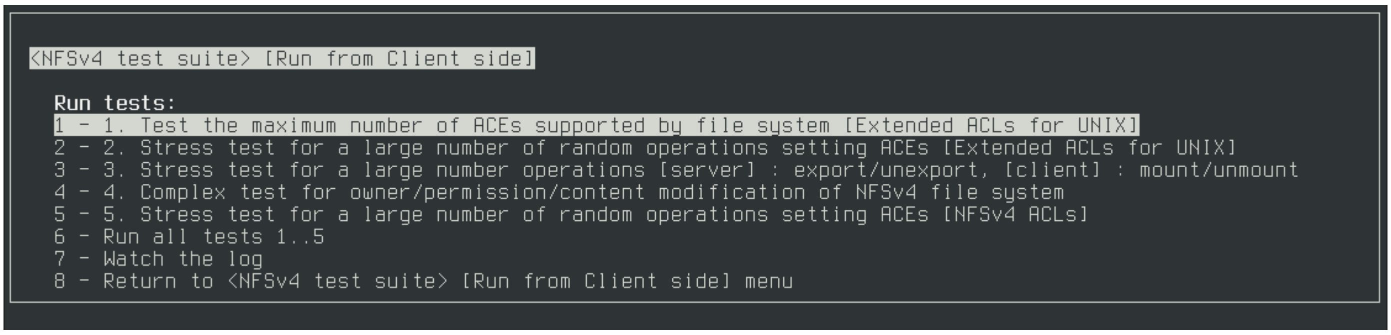

Applications
Places
Thu 17:15
Type to search...
root@fedora: /nfs
'Krusader'
Яндекс.Переводчик – словарь и онлайн перевод на английски...
it
Readme.docx - Writer - Compatibility Mode
NFS4 - [/python/NFS4/NFS4] - .../payload/generator_p.py - Py...
Stress test for a large number operations [server] • - Sublime ...

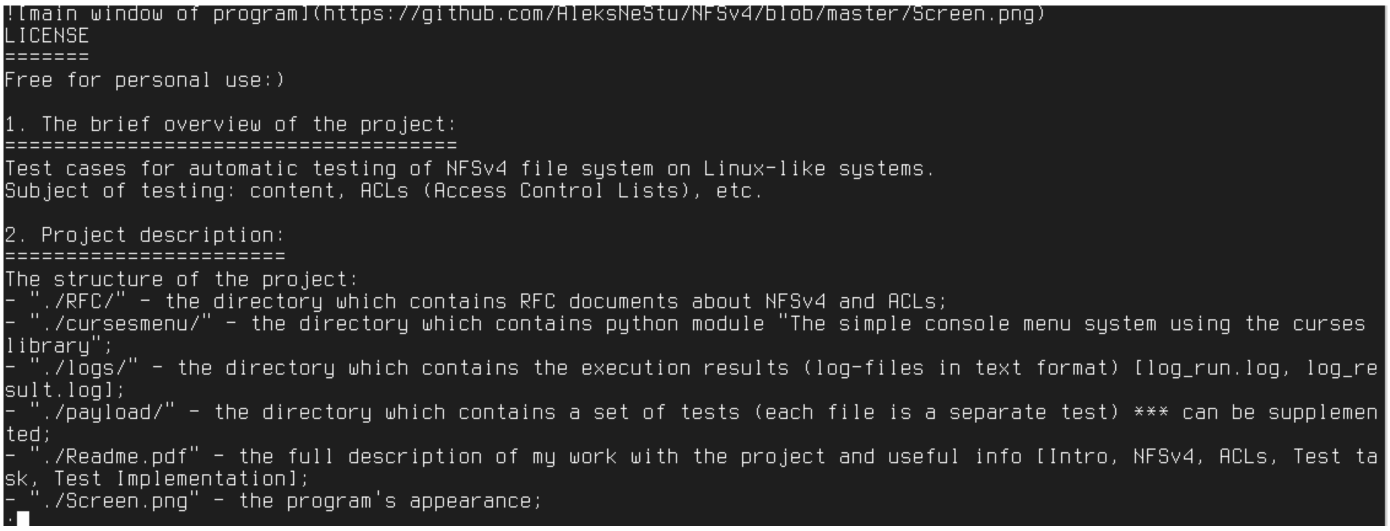


```
import os

class full_generator(object):


#Generate range for groups (group names, gid (7000 + g range) according input data "g - number of groups"
#GID - values between 0 and 999 are typically reserved for system accounts
    def create_groups_n(self, g):
        for i in range(1,g+1):
            gname = "nfs_group" + str(i)
            gid = str(7000 + i)
            self.create_groups(gname, gid)

#Generate groups according the data from "create_groups_n" [consistently in range]
#groupadd -f -g 7000(range) nfs_group(range)
# -f - force	-g GID
    def create_groups(self, gname, gid, log_path="../logs/log_run.log"):
        cmd = commands.getoutput("/usr/sbin/groupadd -f -g " + gid + " " + gname)
        print "     Group: " + gname + " / GID: " + gid + " / has been created"
        if cmd != "":
            print "     Group: " + gname + " / GID: " + gid + " / with errors"
            print cmd

#Generate range for users (user names, uid (7000 + g range) according input data "u - number of users"
#UID - values between 0 and 999 are typically reserved for system accounts
    def create_users_n(self, u):
        # while len(self.groups_list) != options.g: # in order to tun "python generator_p.py -g ** --gg -u **"
        #   print "wait"
        # else:
            for i in range(1,u+1):
                uname = "nfs_user" + str(i)
                uid = str(7000 + i)
                self.create_users(uname, uid)

#Generate users according the data from "create_users_n" [random group select]
#useradd -f -g 7000(range) nfs_group(range)
#UID The numerical value of the user's ID.
#useradd -g "random_from_exist" -m uname -u uid -p nfs
#-g - the group name for a new user's initial group
#-m - create the user's home directory and new user name
#-u - UID
#-p - the encrypted password
    def create_users(self, uname, uid, log_path="../logs/log_run.log"):
```


Applications
Places
Thu 17:35
Type to search...
root@nestu: /home/he
Welcome to Python.org - Google Chrome
/NFS4/temp/GIT/E4 - nfs-test-master/log.txt - Sublime Text
NFS4 - [/python/NFS4/NFS4] - .../payload/generator_p.py - PyCharm 2016.1.4
'Krusader'
/home/he/Pictures/root@nestu: -ho...


```
<NFSv4 test suite> [Run from Client side]

  MENU:
  1 - Read Help
  2 - Check test environment
  3 - Run tests
  4 - Exit
```


```
<NFSv4 test suite> [Run from Client side]

  Run tests:
  1 - 1. Test the maximum number of ACEs supported by file system [Extended ACLs for UNIX]
  2 - 2. Stress test for a large number of random operations setting ACEs [Extended ACLs for UNIX]
  3 - 3. Stress test for a large number operations [server] : export/unexport, [client] : mount/unmount
  4 - 4. Complex test for owner/permission/content modification of NFSv4 file system
  5 - 5. Stress test for a large number of random operations setting ACEs [NFSv4 ACLs]
  6 - Run all tests 1..5
  7 - Watch the log
  8 - Return to <NFSv4 test suite> [Run from Client side] menu
```


```
![main window of program](https://github.com/AleksNeStu/NFSv4/blob/master/Screen.png)
LICENSE
=======
Free for personal use:)

1. The brief overview of the project:
====================================
Test cases for automatic testing of NFSv4 file system on Linux-like systems.
Subject of testing: content, ACLs (Access Control Lists), etc.

2. Project description:
======================
The structure of the project:
- "./RFC/" - the directory which contains RFC documents about NFSv4 and ACLs;
- "./cursesmenu/" - the directory which contains python module "The simple console menu system using the curses
library";
- "./logs/" - the directory which contains the execution results (log-files in text format) [log_run.log, log_re
sult.log];
- "./payload/" - the directory which contains a set of tests (each file is a separate test) *** can be supplemen
ted;
- "./Readme.pdf" - the full description of my work with the project and useful info [Intro, NFSv4, ACLs, Test ta
sk, Test Implementation];
- "./Screen.png" - the program's appearance;
:
```

Test #1: NFSv4 - Test the maximum number of ACEs (Access Control Entries) supported by file system [Extended ACLs for UNIX]

```
========================================================================

Test #1: NFSv4 - Test the maximum number of ACEs (Access Control Entries)
    supported by file system [Extended ACLs for UNIX]

  Limits for the max number of the ACEs per individual file/directory:
  - EXT2, EXT3, EXT4: 32
  - XFS: 25
  - GFS2: 25

  Auto check file system type for NFSv4 server export directory and check
  limits for max count of ACEs [Extended ACLs for UNIX]

========================================================================

In order to run the test enter the required data:

  1) Hostname of the NFSv4 server:

  [input] : rhel
  hostname have resolved
  NFSv4 server:  rhel

  2) Path to the exported directory on NFSv4 server (/nfsdir, /mnt/nfs, ...):

  [input] : /nfs
  NFS server's exported directory is existing
  NFSv4 exported dir:  /nfs

  3) NFSv4 server exported dir filesystem type & ACEs limits:

  NFSv4 server fs type for exp dir  /nfs  :  xfs

  ACEs max count (UNIX extended ACLs):  25
```

```
    4) The number of users and groups to be created on the NFSv4 server:


    The number of users [25..999] [input] : 30

    The number of groups [25..999] [input] : 25


    5) The directory and file which will be created on the export directory on the NFSv4 server:


    Test directory (folder, dir, ...) in NFSv4 server export dir [input] : dir

    Test file (file, goal, ...) in NFSv4 server export dir [input] : file


Great! The input data have been received!

After 5 seconds the test will be started...


=====================================================================


    Client   :  ('Linux 4.4.9-300.fc23.x86_64 x86_64\n', None)

    Hostname :  fedora

    IP       :  192.168.100.182


    Server   :  ('Linux 3.10.0-327.18.2.el7.x86_64 x86_64\n', None)

    Hostname :  rhel

    IP       :  192.168.100.176


    Ping :  fedora  --->>>  rhel

PING rhel (192.168.100.176) 56(84) bytes of data.

64 bytes from rhel (192.168.100.176): icmp_seq=1 ttl=64 time=0.300 ms

64 bytes from rhel (192.168.100.176): icmp_seq=2 ttl=64 time=0.340 ms

64 bytes from rhel (192.168.100.176): icmp_seq=3 ttl=64 time=0.334 ms


--- rhel ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 1999ms

rtt min/avg/max/mdev = 0.300/0.324/0.340/0.027 ms


    [Create 25 groups on the NFSv4 server] :


    Group: nfs_group1 / GID: 7001 / has been created
```

```
Group: nfs_group2 / GID: 7002 / has been created
...
Group: nfs_group25 / GID: 7025 / has been created


[Create 30 users on the NFSv4 server] :


User: nfs_user1 / UID: 7001 / GID: nfs_group20 / has been created
User: nfs_user2 / UID: 7002 / GID: nfs_group12 / has been created
...
User: nfs_user28 / UID: 7028 / GID: nfs_group18 / has been created
User: nfs_user29 / UID: 7029 / GID: nfs_group6 / has been created
User: nfs_user30 / UID: 7030 / GID: nfs_group24 / has been created


[Create the directory and file on the NFSv4 server export directory] :


Test directory on the [rhel] : /nfs/dir has been created
Test file on the [rhel] : /nfs/file has been created


[Test the maximum number of ACEs for directory] :


ACE # 5 : setfacl -m u:nfs_user24:x /nfs/dir
ACE # 6 : setfacl -m u:nfs_user26:r /nfs/dir
ACE # 7 : setfacl -m u:nfs_user21:x /nfs/dir
ACE # 8 : setfacl -m u:nfs_user4:xr /nfs/dir
ACE # 9 : setfacl -m u:nfs_user28:w /nfs/dir
ACE # 10 : setfacl -m u:nfs_user18:xr /nfs/dir
ACE # 11 : setfacl -m u:nfs_user12:xr /nfs/dir
ACE # 12 : setfacl -m u:nfs_user20:r /nfs/dir
ACE # 13 : setfacl -m u:nfs_user3:xr /nfs/dir
ACE # 14 : setfacl -m u:nfs_user2:x /nfs/dir
ACE # 15 : setfacl -m u:nfs_user9:w /nfs/dir
ACE # 16 : setfacl -m u:nfs_user8:xr /nfs/dir
ACE # 17 : setfacl -m u:nfs_user11:wr /nfs/dir
ACE # 18 : setfacl -m u:nfs_user10:xr /nfs/dir
ACE # 19 : setfacl -m u:nfs_user27:wr /nfs/dir
ACE # 20 : setfacl -m u:nfs_user16:wr /nfs/dir
ACE # 21 : setfacl -m u:nfs_user1:xr /nfs/dir
```

```
    ACE # 22 : setfacl -m u:nfs_user7:xr /nfs/dir

    ACE # 23 : setfacl -m u:nfs_user14:xwr /nfs/dir

    ACE # 24 : setfacl -m u:nfs_user15:x /nfs/dir

    ACE # 25 : setfacl -m u:nfs_user30:xr /nfs/dir

    ACE # 26 : setfacl -m u:nfs_user5:wr /nfs/dir - ERROR!!!

setfacl: /nfs/dir: Argument list too long

    Reached the maximum number of ACEs: 25

    THE TEST #1 HAS BEEN PASSED!!!


    [Test the maximum number of ACEs for file] :


    ACE # 5 : setfacl -m u:nfs_user3:r /nfs/file

    ACE # 6 : setfacl -m u:nfs_user21:wr /nfs/file

    ACE # 7 : setfacl -m u:nfs_user28:xr /nfs/file

    ACE # 8 : setfacl -m u:nfs_user29:r /nfs/file

    ACE # 9 : setfacl -m u:nfs_user27:wr /nfs/file

    ACE # 10 : setfacl -m u:nfs_user15:wr /nfs/file

    ACE # 11 : setfacl -m u:nfs_user20:w /nfs/file

    ACE # 12 : setfacl -m u:nfs_user12:x /nfs/file

    ACE # 13 : setfacl -m u:nfs_user25:xwr /nfs/file

    ACE # 14 : setfacl -m u:nfs_user7:r /nfs/file

    ACE # 15 : setfacl -m u:nfs_user13:xwr /nfs/file

    ACE # 16 : setfacl -m u:nfs_user30:wr /nfs/file

    ACE # 17 : setfacl -m u:nfs_user9:xr /nfs/file

    ACE # 18 : setfacl -m u:nfs_user18:w /nfs/file

    ACE # 19 : setfacl -m u:nfs_user2:xwr /nfs/file

    ACE # 20 : setfacl -m u:nfs_user16:xwr /nfs/file

    ACE # 21 : setfacl -m u:nfs_user10:wr /nfs/file

    ACE # 22 : setfacl -m u:nfs_user1:wr /nfs/file

    ACE # 23 : setfacl -m u:nfs_user23:wr /nfs/file

    ACE # 24 : setfacl -m u:nfs_user19:wr /nfs/file

    ACE # 25 : setfacl -m u:nfs_user8:xr /nfs/file

    ACE # 26 : setfacl -m u:nfs_user5:x /nfs/file - ERROR!!!

setfacl: /nfs/file: Argument list too long

    Reached the maximum number of ACEs: 25

    THE TEST #1 HAS BEEN PASSED!!!
```

```
   [Clean created directory and file on the NFSv4 server] :


   Test directory on the [rhel] :  /nfs/dir  has been removed
   Test file on the [rhel] :  /nfs/file  has been removed


   [Clean created users and groups on the NFSv4 server] :


   User del: nfs_user1 / has been done
   ...
   User del: nfs_user30 / has been done
   Group del: nfs_group1 / has been done
   ...
   Group del: nfs_group24 / has been done
   Group del: nfs_group25 / has been done



##################################################################################


Test #1 NFSv4 maximum number of ACEs [Extended ACLs for UNIX]


     [PASSED] [PASSED] [PASSED] [PASSED] [PASSED] [PASSED] [PASSED]


   In order to get more information:
   ./logs/log_run.log" - execution log (detailed information)
   ./logs/log_result.log - log with the results


##################################################################################
```

Test #2 NFSv4 - Stress test for a large number of random operations setting ACEs [Extended ACLs for UNIX] [PASSED]

```
===============================================================================

Test #2: NFSv4 - Stress test for a large number of random operations
       setting ACEs [Extended ACLs for UNIX]

   The testing of correct operation of the services of NFSv4 server
   with a large number of random operations setting ACEs
   (Access Control Entries)

   Part 1 - Emulate a large number random operations with ACEs for export
          directory on the NFSv4 server, health check NFSv4 service

   Part 2 - Emulate a large number random operations with ACEs for random
          files in export directory on the NFSv4 server, health check
          NFSv4 service

===============================================================================

In order to run the test enter the required data:

   1) Hostname of the NFSv4 server:

   [input] : rhel
   hostname have resolved
   NFSv4 server:  rhel

   2) Path to the exported directory on NFSv4 server (/nfsdir, /mnt/nfs, ...):

   [input] : /nfs
   NFS server's exported directory is existing
   NFSv4 exported dir:  /nfs

   3) The number of users and groups to be created on the NFSv4 server:
    The number of users [50..1000] [input] : 50
    The number of groups [50..1000] [input] : 60


    4) The number of files to be created in the export dir on the NFSv4 server:


    The number of files [10..50] [input] : 20


    5) The number of cycles to perform random operation in the test:


    The number of cycles [500..10000] [input] : 500


Great! The input data have been received!
After 5 seconds the test will be started...


=====================================================================


    Client   :  ('Linux 4.4.9-300.fc23.x86_64 x86_64\n', None)
    Hostname :  fedora
    IP       :  192.168.100.182


    Server   :  ('Linux 3.10.0-327.18.2.el7.x86_64 x86_64\n', None)
    Hostname :  rhel
    IP       :  192.168.100.176


    Ping :  fedora  --->>>  rhel
PING rhel (192.168.100.176) 56(84) bytes of data.
64 bytes from rhel (192.168.100.176): icmp_seq=1 ttl=64 time=0.276 ms
64 bytes from rhel (192.168.100.176): icmp_seq=2 ttl=64 time=0.361 ms
64 bytes from rhel (192.168.100.176): icmp_seq=3 ttl=64 time=0.365 ms


--- rhel ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.276/0.334/0.365/0.041 ms


    [Create 60 groups on the NFSv4 server] :


    Group: nfs_group1 / GID: 7001 / has been created
```

```
   ...
   Group: nfs_group60 / GID: 7060 / has been created

   [Create 50 users on the NFSv4 server] :

   User: nfs_user1 / UID: 7001 / GID: nfs_group56 / has been created
   ...
   User: nfs_user50 / UID: 7050 / GID: nfs_group11 / has been created

   [Part 1 - Stress test for a large number random ACEs for export directory on the NFSv4 server] :

   ACE # 1 : setfacl -m u:nfs_user21:wr /nfs & setfacl -x u:nfs_user21 /nfs
   ...
   ACE # 497 : setfacl -m u:nfs_user8:xwr /nfs & setfacl -x u:nfs_user8 /nfs
   ACE # 498 : setfacl -m u:nfs_user44:xw /nfs & setfacl -x u:nfs_user44 /nfs
   ACE # 499 : setfacl -m u:nfs_user28:r /nfs & setfacl -x u:nfs_user28 /nfs
   ACE # 500 : setfacl -m u:nfs_user11:xr /nfs & setfacl -x u:nfs_user11 /nfs

   THE TEST #2 PART 1 HAS BEEN PASSED!!!

   [Part 2 - Stress test for a large number random ACEs for random files in export directory on the NFSv4 server] :



   File: /nfs/nfs_file1 / has been created
   File: /nfs/nfs_file1 / with errors
touch: cannot touch '/nfs/nfs_file1': No such file or directory
   File: /nfs/nfs_file2 / has been created
   File: /nfs/nfs_file2 / with errors
touch: cannot touch '/nfs/nfs_file2': No such file or directory
   ...
   File: /nfs/nfs_file20 / has been created
   File: /nfs/nfs_file20 / with errors
touch: cannot touch '/nfs/nfs_file20': No such file or directory

   ACE # 1 : setfacl -x u:nfs_user14 /nfs/nfs_file12
setfacl: /nfs/nfs_file12: No such file or directory
   ...
```

```
    ACE # 498 : setfacl -x u:nfs_user16 /nfs/nfs_file3
setfacl: /nfs/nfs_file3: No such file or directory
    ACE # 499 : setfacl -m g:nfs_group51:xw /nfs/nfs_file4
setfacl: /nfs/nfs_file4: No such file or directory
    ACE # 500 : setfacl -m g:nfs_group14:xw /nfs/nfs_file2
setfacl: /nfs/nfs_file2: No such file or directory

    [Check the status of a NFS service on the NFSv4 server after stress test] :

● nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; vendor preset: disabled)
   Active: active (exited) since Thu 2016-06-02 18:46:57 MSK; 9min ago
  Process: 994 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited, status=0/SUCCESS)
  Process: 993 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
 Main PID: 994 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/nfs-server.service

Jun 02 18:46:55 rhel systemd[1]: Starting NFS server and services...
Jun 02 18:46:57 rhel systemd[1]: Started NFS server and services.

    THE TEST #2 PART 2 HAS BEEN PASSED!!!

    [Clean created test data from the NFSv4 server] :

    Created ACEs have been deleted from export dir and files in it on the NFS server.
    Created files have been deleted from export dir on the NFS server.
    Created users and groups have been deleted from the NFS server.


################################################################################

Test #2 NFSv4 - Stress test for a large number of random operations
        setting ACEs [Extended ACLs for UNIX] [PASSED]

        [PASSED] [PASSED] [PASSED] [PASSED] [PASSED] [PASSED] [PASSED]

    In order to get more information:
```

./logs/log_run.log" - execution log (detailed information)

./logs/log_result.log - log with the results

###

Test #3: NFSv4 - Stress test for a large number operations:
[NFSv4 server side] : export / unexport
[NFSv4 client side] : mount / unmount

```
========================================================================

Test #3: NFSv4 - Stress test for a large number operations:
    [NFSv4 server side] : export / unexport
    [NFSv4 client side] : mount / unmount

    Part 1 - Emulation of export a large number of directories on the
        NFSv4 server and their subsequent mounting on the NFSv4
        client, health check NFSv4 service

    Part 2 - Emulation of a large number of random operations with
        exports on NFSv4 server and mounts on NFSv4 client,
        health check NFSv4 service

    *** The test data will be created in the directory /mnt/* from
    both sides NFSv4 client and server

========================================================================

In order to run the test enter the required data:

  1) Hostname of the NFSv4 server:

  [input] : rhel
  hostname have resolved
  NFSv4 server:  rhel

  2) The number of exported (mounted) directories:

  The number of directories [500..10000] [input] : 500

  3) The number of random operations: export/unexport (server), mount/unmount (client):
    The number of random operations [1000..20000] [input] : 1000

Great! The input data have been received!
After 5 seconds the test will be started...

==========================================================================

    Client   :  ('Linux 4.4.9-300.fc23.x86_64 x86_64\n', None)
    Hostname :  fedora
    IP       :  192.168.100.182

    Server   :  ('Linux 3.10.0-327.18.2.el7.x86_64 x86_64\n', None)
    Hostname :  rhel
    IP       :  192.168.100.176

    Ping :  fedora  --->>>  rhel
PING rhel (192.168.100.176) 56(84) bytes of data.
64 bytes from rhel (192.168.100.176): icmp_seq=1 ttl=64 time=0.374 ms
64 bytes from rhel (192.168.100.176): icmp_seq=2 ttl=64 time=0.395 ms
64 bytes from rhel (192.168.100.176): icmp_seq=3 ttl=64 time=0.313 ms

--- rhel ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.313/0.360/0.395/0.041 ms



    [Part 1 - Stress test for export a large number of directories on the NFSv4 server
            and their subsequent mounting on the NFSv4 client] :

    [Create 500 directories (both), export them (server) and mount them (client)] :

    #1 STEP
    [Server] : directory /mnt/nfs_exp1 has been created
    [Server] : directory /mnt/nfs_exp1 has been exported
    [Client] : directory /mnt/nfs_mnt1 has been created
    [Client] : directory /mnt/nfs_exp1 has been mounted to local /mnt/nfs_mnt1
    ...
```

```
   #499 STEP
   [Server] : directory /mnt/nfs_exp499 has been created
   [Server] : directory /mnt/nfs_exp499 created errors
poll: protocol failure in circuit setup
   [Server] : directory /mnt/nfs_exp499 has been exported
   [Server] : directory /mnt/nfs_exp499 exported errors
poll: protocol failure in circuit setup
   [Client] : directory /mnt/nfs_mnt499 has been created
   [Client] : directory /mnt/nfs_exp499 has been mounted to local /mnt/nfs_mnt499
   [Client] : directory /mnt/nfs_exp499 mounted errors
mount.nfs4: access denied by server while mounting rhel:/mnt/nfs_exp499
   #500 STEP
   [Server] : directory /mnt/nfs_exp500 has been created
   [Server] : directory /mnt/nfs_exp500 created errors
poll: protocol failure in circuit setup
   [Server] : directory /mnt/nfs_exp500 has been exported
   [Server] : directory /mnt/nfs_exp500 exported errors
poll: protocol failure in circuit setup
   [Client] : directory /mnt/nfs_mnt500 has been created
   [Client] : directory /mnt/nfs_exp500 has been mounted to local /mnt/nfs_mnt500
   [Client] : directory /mnt/nfs_exp500 mounted errors
mount.nfs4: access denied by server while mounting rhel:/mnt/nfs_exp500

   [Check the final export directory on the server] :

   [Check the final mount directory on the client] :

   THE TEST #3 PART 1 HAS BEEN FAILED!!!

   [Part 2 - Stress test for large number random operations: exports:
              export/unexport (NFSv4 server), mount/unmount (NFSv4 client)] :

   [Execute random operations with 1000 exports: export/unexport (server), mount/unmount (client)] :

   #1 STEP [Server] : directory /mnt/nfs_exp1 has been exported
   #2 STEP [Server] : directory /mnt/nfs_exp2 has been unexported
  ...
```

```
    #997 STEP [Server] : directory /mnt/nfs_exp997 has been exported
    #997 STEP [Server] : directory /mnt/nfs_exp997 exported errors
exportfs: Failed to stat /mnt/nfs_exp997: No such file or directory
    #998 STEP [Client] : directory /mnt/nfs_exp998 has been mounted to local /mnt/nfs_mnt998
    #998 STEP [Client] : directory /mnt/nfs_exp998 mounted errors
mount.nfs4: mount point /mnt/nfs_mnt998 does not exist
    #999 STEP [Client] : directory /mnt/nfs_mnt999 has been unmounted
    #999 STEP [Client] : directory /mnt/nfs_mnt999 unmounted errors
umount: /mnt/nfs_mnt999: mountpoint not found
    #1000 STEP [Server] : directory /mnt/nfs_exp1000 has been exported
    #1000 STEP [Server] : directory /mnt/nfs_exp1000 exported errors
exportfs: Failed to stat /mnt/nfs_exp1000: No such file or directory

    [Check the status of a NFS service on the NFSv4 server after stress test] :

● nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; vendor preset: disabled)
   Active: active (exited) since Thu 2016-06-02 18:46:57 MSK; 23min ago
  Process: 994 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited, status=0/SUCCESS)
  Process: 993 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
 Main PID: 994 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/nfs-server.service

Jun 02 18:46:55 rhel systemd[1]: Starting NFS server and services...
Jun 02 18:46:57 rhel systemd[1]: Started NFS server and services.

    THE TEST #3 PART 2 HAS BEEN PASSED!!!



    [Clean created test data from the NFSv4 client and server] :

    [Client] : Mounted directories have been unmonted on NFSv4 client.
umount: /mnt/nfs_mnt100: not mounted
...
umount: /mnt/nfs_mnt9: not mounted
umount: /mnt/nfs_mnt97: not mounted
    [Client] : Old mounted directories have been deleted from NFSv4 client.
   [Server] : Exported directories have been unexported on the NFSv4 server.
exportfs: Failed to stat /nfs: No such file or directory
   [Client] : Old exported directories have been deleted from NFSv4 server.

################################################################################

Test #3 NFSv4 - Stress test for a large number operations:
      [NFSv4 server side] : export / unexport
      [NFSv4 client side] : mount / unmount

      [PASSED] [PASSED] [PASSED] [PASSED] [PASSED] [PASSED] [PASSED]

   In order to get more information:
   ./logs/log_run.log" - execution log (detailed information)
   ./logs/log_result.log - log with the results

################################################################################
```

Test #4: The complex test for owner/permission/content modification of NFSv4 file system [POSIX]

```
Test #4: The complex test for owner/permission/content modification of
    NFSv4 file system [POSIX]

    Part 1 - Check the ability of check the permissions of the files

    > Description    : Check the ability of changes the permissions of the files
    > Steps          : Run used functions of full_generator
    > Expected result : Display "TEST #4 PART 2 HAS BEEN PASSED"

    Part 2 - Check the ability of check the permissions of the directories

    > Description    : Check the ability of changes the permissions of the directories
    > Steps          : Run used functions of full_generator
    > Expected result : Display "TEST #4 PART 2 HAS BEEN PASSED"

    *** All parts of the complex test will be performed sequentially
    and will be running in the mount directory on NFSv4 client side

================================================================================

In order to run the test enter the required data:

  1) Hostname of the NFSv4 server:

  [input] : rhel
  hostname have resolved
  NFSv4 server:  rhel

  2) Path to the exported (test) directory on NFSv4 server (/nfsdir, /mnt/nfs, ...):

  [input] : /nfs
  NFS server's exported directory is existing
  NFSv4 exported dir:  /nfs
```

```
    3) The number files, directories to be created on the NFSv4 client in mount dir:

    The number of objects [5..50] [input] : 10

Great! The input data have been received!
After 5 seconds the test will be started...

==========================================================================================

    Client   :  ('Linux 4.4.9-300.fc23.x86_64 x86_64\n', None)
    Hostname :  fedora
    IP       :  192.168.100.182

    Server   :  ('Linux 3.10.0-327.18.2.el7.x86_64 x86_64\n', None)
    Hostname :  rhel
    IP       :  192.168.100.176

    Ping :  fedora  --->>>  rhel
PING rhel (192.168.100.176) 56(84) bytes of data.
64 bytes from rhel (192.168.100.176): icmp_seq=1 ttl=64 time=0.271 ms
64 bytes from rhel (192.168.100.176): icmp_seq=2 ttl=64 time=0.300 ms
64 bytes from rhel (192.168.100.176): icmp_seq=3 ttl=64 time=0.310 ms

--- rhel ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.271/0.293/0.310/0.025 ms

    [Create 10  files, directories (random chmod) on the NFSv4 client in mount dir] :

    File: /nfs/nfs_file1 / has been created with permissions: u=w,g=wr,o=xr
    File: /nfs/nfs_file2 / has been created with permissions: u=xr,g=xwr,o=x
    File: /nfs/nfs_file3 / has been created with permissions: u=xw,g=w,o=xw
    File: /nfs/nfs_file4 / has been created with permissions: u=xw,g=x,o=xw
    File: /nfs/nfs_file5 / has been created with permissions: u=xr,g=r,o=x
    File: /nfs/nfs_file6 / has been created with permissions: u=w,g=xr,o=xwr
    File: /nfs/nfs_file7 / has been created with permissions: u=w,g=r,o=r
    File: /nfs/nfs_file8 / has been created with permissions: u=xwr,g=xw,o=xwr
```

```
File: /nfs/nfs_file9 / has been created with permissions: u=xw,g=r,o=xwr

File: /nfs/nfs_file10 / has been created with permissions: u=r,g=x,o=wr


Directory: /nfs/nfs_dir1 / has been created with permissions: u=w,g=xwr,o=r

Directory: /nfs/nfs_dir2 / has been created with permissions: u=r,g=wr,o=xwr

Directory: /nfs/nfs_dir3 / has been created with permissions: u=xr,g=xw,o=x

Directory: /nfs/nfs_dir4 / has been created with permissions: u=xr,g=x,o=x

Directory: /nfs/nfs_dir5 / has been created with permissions: u=xwr,g=xwr,o=r

Directory: /nfs/nfs_dir6 / has been created with permissions: u=xwr,g=xw,o=r

Directory: /nfs/nfs_dir7 / has been created with permissions: u=xw,g=xwr,o=xwr

Directory: /nfs/nfs_dir8 / has been created with permissions: u=xwr,g=x,o=xwr

Directory: /nfs/nfs_dir9 / has been created with permissions: u=w,g=x,o=xwr

Directory: /nfs/nfs_dir10 / has been created with permissions: u=xwr,g=xr,o=r



[Part 1 - Check the ability of check the permissions of the files] :



[Get directories and files in test directory /nfs] :


Get dir: /nfs/nfs_dir_444 permissions: 444

Get dir: /nfs/nfs_dir_666 permissions: 666

Get dir: /nfs/nfs_dir1 permissions: 274

Get dir: /nfs/nfs_dir2 permissions: 467

Get dir: /nfs/nfs_dir3 permissions: 531

Get dir: /nfs/nfs_dir4 permissions: 511

Get dir: /nfs/nfs_dir5 permissions: 774

Get dir: /nfs/nfs_dir6 permissions: 734

Get dir: /nfs/nfs_dir7 permissions: 377

Get dir: /nfs/nfs_dir8 permissions: 717

Get dir: /nfs/nfs_dir9 permissions: 217

Get dir: /nfs/nfs_dir10 permissions: 754

Get file: /nfs/nfs_file_444 permissions: 444

Get file: /nfs/nfs_file_666 permissions: 666

Get file: /nfs/nfs_file1 permissions: 265

Get file: /nfs/nfs_file2 permissions: 571

Get file: /nfs/nfs_file3 permissions: 323
Get file: /nfs/nfs_file4 permissions: 313

Get file: /nfs/nfs_file5 permissions: 541

Get file: /nfs/nfs_file6 permissions: 257

Get file: /nfs/nfs_file7 permissions: 244

Get file: /nfs/nfs_file8 permissions: 737

Get file: /nfs/nfs_file9 permissions: 347

Get file: /nfs/nfs_file10 permissions: 416


[Check the permissions of the files in test directory /nfs ] :


[Test to read for files with permissions: r] / Expected: PASSED


Try reading the file /nfs/nfs_file_444 / Expected: True

Try reading the file /nfs/nfs_file_444 / Result: True

Try writing <NFSv4 test> to the file /nfs/nfs_file_444 / Expected: False

Try writing <NFSv4 test> to the file /nfs/nfs_file_444 / Result: True


[Test to read for files with permissions: r] / Result: PASSED



[Test to read and write for files with permissions: rw] / Expected: PASSED


Try reading the file /nfs/nfs_file_666 / Expected: True

Try reading the file /nfs/nfs_file_666 / Result: True

Try writing <NFSv4 test> to the file /nfs/nfs_file_666 / Expected: True

Try writing <NFSv4 test> to the file /nfs/nfs_file_666 / Result: True

Try reading the file /nfs/nfs_file8 / Expected: True

Try reading the file /nfs/nfs_file8 / Result: True

Try writing <NFSv4 test> to the file /nfs/nfs_file8 / Expected: True

Try writing <NFSv4 test> to the file /nfs/nfs_file8 / Result: True


[Test to read and write for files with permissions: rw] / Result: PASSED



THE TEST #4 PART 1 HAS BEEN PASSED!!!


[Part 2 - Check the ability of check the permissions of the directories] :
```

```
[Get directories and files in test directory /nfs] :

Get dir: /nfs/nfs_dir_444 permissions: 444
Get dir: /nfs/nfs_dir_666 permissions: 666
Get dir: /nfs/nfs_dir1 permissions: 274
Get dir: /nfs/nfs_dir2 permissions: 467
Get dir: /nfs/nfs_dir3 permissions: 531
Get dir: /nfs/nfs_dir4 permissions: 511
Get dir: /nfs/nfs_dir5 permissions: 774
Get dir: /nfs/nfs_dir6 permissions: 734
Get dir: /nfs/nfs_dir7 permissions: 377
Get dir: /nfs/nfs_dir8 permissions: 717
Get dir: /nfs/nfs_dir9 permissions: 217
Get dir: /nfs/nfs_dir10 permissions: 754
Get file: /nfs/nfs_file_444 permissions: 444
Get file: /nfs/nfs_file_666 permissions: 666
Get file: /nfs/nfs_file1 permissions: 265
Get file: /nfs/nfs_file2 permissions: 571
Get file: /nfs/nfs_file3 permissions: 323
Get file: /nfs/nfs_file4 permissions: 313
Get file: /nfs/nfs_file5 permissions: 541
Get file: /nfs/nfs_file6 permissions: 257
Get file: /nfs/nfs_file7 permissions: 244
Get file: /nfs/nfs_file8 permissions: 737
Get file: /nfs/nfs_file9 permissions: 347
Get file: /nfs/nfs_file10 permissions: 416

[Check the permissions of the dirs in test directory /nfs ] :

[Test to read for dirs with permissions: r] / Expected: PASSED

Try reading the dir /nfs/nfs_dir_444 / Expected: True
Try reading the dir /nfs/nfs_dir_444 / Result: True
Try reading the dir /nfs/nfs_dir_444 / Expected: True
Try reading the dir /nfs/nfs_dir_444 / Result: True
```

```
    [Test to read for dirs with permissions: r] / Result: PASSED



    THE TEST #4 PART 2 HAS BEEN PASSED!!!



    [Clean created test data from the NFSv4 client]

    [Created test data have been cleaned from the NFSv4 client]



    [Get directories and files in test directory /nfs] :



    [Check the permissions of the dirs in test directory /nfs ] :

    [Test to read for dirs with permissions: r] / Expected: PASSED


    Try reading the dir /nfs/nfs_dir_444 / Expected: True
    Try reading the dir /nfs/nfs_dir_444 / Result: False
    Try reading the dir /nfs/nfs_dir_444 / Expected: True
    Try reading the dir /nfs/nfs_dir_444 / Result: False

    [Test to read for dirs with permissions: r] / Result: FAILED



    THE TEST #4 PART 2 HAS BEEN FAILED!!!

########################################################################################

Test #4 The complex test for owner/permission/content modification of
      NFSv4 file system

      [FAILED] [FAILED] [FAILED] [FAILED] [FAILED] [FAILED] [FAILED]

    In order to get more information:
```

./logs/log_run.log" - execution log (detailed information)

./logs/log_result.log - log with the results

##

Test #5: Stress test for a large number of random operations setting ACEs (Access Control Entries) [NFSv4 ACLs]

Coming soon!!!