Testing iSCSI/SCSI Protocol Compliance Using Libiscsi

Ronnie Sahlberg
Google

SCSI Testing + Libiscsi

Ronnie Sahlberg
Google

Who Am I ?

Ronnie Sahlberg.
I work for Google.
I work on open source in my spare time.

Wireshark, TGTD, NFS, iSCSI, CTDB, ...

What will I cover in this presentation?

Libiscsi, what, why?
Libiscsi API
The test suite that comes with it
Demo

What is libiscsi?

What is libiscsi

Userspace iscsi initiator.
High performance.
Mature.
Permissive licence.
Highly portable.
    Runs natively on Linux, Solaris, most Unices, OSX, Windows.

Why?

Originally developed to provide builtin iSCSI support for KVM/QEMU.
Process private LUNs.
No need to be root to access the device.

Since transformed to also contain a SCSI test suite.

Where?

Github:
https://github.com/sahlberg/libiscsi

Comes prepackaged on most linux distributions.

Licence

LGPL for the library,

GPL for the test suite and utilities.

URL Syntax

iSCSI Portal URL format :
iscsi://[<username>[%<password>]@]<host>[:<port>]

iSCSI LUN URL format :
iscsi://
[<username>[%<password>]@]<host>[:<port>]/<target-iqn>/<lun>

URL Syntax

```
$ iscsi-ls --show-luns iscsi://10.10.10.10
Target:iqn.ronnie.test Portal:10.10.10.10:3260,1
Lun:0    Type:STORAGE_ARRAY_CONTROLLER
Lun:1    Type:DIRECT_ACCESS (Size:1023M)

$ sg_inq iscsi://10.10.10.10/iqn.ronnie.test/1
standard INQUIRY:
  PQual=0  Device_type=0  RMB=0  version=0x05
[SPC-3]
...
```

Libiscsi API

Libiscsi API

Sync API with support for most SBC commands :

```
EXTERN struct scsi_task *
iscsi_writesame16_sync(struct iscsi_context *iscsi, int lun, uint64_t lba,
                       unsigned char *data, uint32_t datalen,
                       uint32_t num_blocks,
                       int anchor, int unmap, int wrprotect, int group);
```

Fully nonblocking async API :

```
EXTERN struct scsi_task *
iscsi_writesame16_task(struct iscsi_context *iscsi, int lun, uint64_t lba,
                       unsigned char *data, uint32_t datalen,
                       uint32_t num_blocks,
                       int anchor, int unmap, int wrprotect, int group,
                       iscsi_command_cb cb, void *private_data)
```

API Documentation

API documented in iscsi.h + scsi-lowlevel.h

Plenty of examples provided for both sync and async API.

The test suite

Why a testsuite?

There wasn't one available.
Was interesting and useful while developing for STGT and Wireshark.

Tests on various targets show there is need/value to provide a test suite.

iscsi-test-cu

Test suite based on libCUNIT
Mainly SBC tests but some iSCSI tests too.

Need/want to expand with more iSCSI and more commandsets.

Replaces a previous ad-hoc test suite called iscsi-test.

Iscsi-test-cu --help

```
  -i|--initiator-name=iqn-name     Initiatorname to use [iqn.2007-
10.com.github:sahlberg:libiscsi:iscsi-test]
  -I|--initiator-name-2=iqn-name   2nd Initiatorname to use [iqn.2007-
10.com.github:sahlberg:libiscsi:iscsi-test-2]
  -t|--test=test-name-reg-exp
  -l|--list                    List all tests and exit
  -d|--dataloss                Allow destructive tests
  -S|--allow-sanitize          Allow sanitize-opcode tests
  -g|--ignore                  Error Action: Ignore test errors [DEFAULT]
  -f|--fail                    Error Action: FAIL if any tests fail
  -A|--abort                   Error Action: ABORT if any tests fail
  -u|--usb                     The device is attached to a USB bus.
                               Additional restrictions apply, such as maximum transfer length
120kb.
  -s|--silent                  Test Mode: Silent
  -n|--normal                  Test Mode: Normal
  -v|--verbose                 Test Mode: Verbose [DEFAULT]
  -V|--Verbose-scsi            Enable verbose SCSI logging [default SILENT]
```

Family.Suite.Test

Tests are divided in a hierarchy of Family.Suite.Test

We currently have 173 individual tests.

Family is divided up in areas such as :
ALL
SCSI
iSCSI
SCSI-USB-SBC

Some suites/tests are part of multiple suites.

Suites

Suites define specific areas to test.
For example one suite for each SCSI command.
SCSI.READ10.*

But can also be a collection of specific type of tests:
ISCSI.ISCSIResiduals.*
That verifies that a target returns iSCSI underflow/overflow for various commands.

Tests

Tests are used for testing small contained types of behaviour for a specific suite.

SCSI.Write16.BeyondEol

Which performs various different tests to validate that a target returns correct sense when WRITE16 commands are accessing medium beyond the end of the device.

List of all tests

<demo>
Iscsi-test-cu –list | less
And get an overviews of what tests exist
</demo>

Tests are self-documenting

-V will show all operations a test performs and what the expected result is.

<demo>
Iscsi-test-cu iscsi://10.10.10.10/iqn.ronnie.test/1 –test SCSI.Read10.ReadProtect -V
</demo>

--dataloss

Tests will not write to the media unless –dataloss is specified.

But don't count on this. There could be bugs.

Tests for ThinProvisioning

This is a reasonably new addition to SBC so there are few existing tests for it.

We have test suites for GETLBASTATUS, UNMAP and both WRITESAME commands.

SCSI.WriteSame16.Unmap

<demo>
iscsi-test-cu iscsi://10.10.10.10/iqn.ronnie.test/1 --test SCSI.WriteSame16.Unmap -V --dataloss
</demo>

SCSI.WriteSame16.UnmapVPD

<demo>
iscsi-test-cu iscsi://10.10.10.10/iqn.ronnie.test/1 --test SCSI.WriteSame16.UnmapVPD -V --dataloss
</demo>

Look at the code for Unmap/UnmapVPD

<demo>
Test-tool/test_writesame16_unmap.c
Test-tool/test_writesame16_unmap_vpd.c
</demo>

Tests for removable devices and readonly devices

PAMR devices can not be ejected.
PAMR is reset for proper TaskMGMT functions

Medium writes fail with proper sense codes on readonly devices.

<demo>
Prevent allow 2 nexuses
</demo>

64 bit LUN overflow.

Very common that targets get wrong.
Large LBA + TL can wrap and cause LBA+TD < device-size tests to fail.

Imagine a soft target that fails the test and that does not use/support sparse files.
What will happen when you start writing to the LUN file at offset ~2^64 ? Unfun things perhaps ?

Invalid commands

iSCSI.iSCSIResiduals.Read10Invalid
What about a iSCSI command with the W flag set but the CDB is READ10 ?

<look at this in wireshark>
What shopuld happen ? Both underflow and overflow at the same time? Discuss.

What should not happen is kernel panic, iSCSI REJECT or similar.

What now?

The test suite is the most comprehensive public one available.

It is easy to extend.

And from all my testing I really think we can all benefit from a good standard test suite.

Anyone interested in using the testsuite and maybe contribute to new tests?

Q n A