Samba 4 Testing Improvements

Jelmer Vernooij

Samba Team SerNet GmbH

Samba Developer Meeting 2007

Agenda

1. **Testing a year ago**
   - Testing a year ago
2. **Infrastructure improvements**
   - New selftest code
   - Subunit
   - The smbtorture C API
3. **New tests**
   - Local tests
   - spoolss callback testing
4. **Where to from here?**
   - Porting to Samba 3
   - Interoperability testing
   - Deployment testing
   - Benchmark tests in the buildfarm
   - More tests!

What sort of testing does Samba already do?

- Regression tests
- Benchmarks
- Interoperability tests

Only regression tests run by the buildfarm

Overview

selftest

1. determines available tests (as output by *tests_all.sh*)
2. determines which tests to run
3. runs tests and parses output
4. prints summary

A test

- Command to run (usually *smbtorture*), environment and description
- Should return zero on success, non-zero on failure
- Can optionally output “Subunit”

New selftest code

- Runs individual tests (see unittesting)
  make test TESTS=regex
- Code coverage output
- Improved error reporting
- HTML Output
- Clean support for multiple targets: *Samba4*, *Samba3*, *Windows*, ...

Unit testing

- Divide up tests into small units

- xUnit API in common use in other projects (pyunit, nUnit, JUnit, . . .)
- Test-driven development
  - Saves time
  - Refactoring with confidence

- Errors and failures are different:

failures an explicit check failed (e.g. function returned 1 where 0 was expected)

errors unexpectedly aborted for some reason (usually segfault)

Unit testing

- Divide up tests into small units
  - Ability to run an individual test
  - Ability to skip or mark as failure small tests
- xUnit API in common use in other projects (pyunit, nUnit, JUnit, . . .)
- Test-driven development
  - Saves time
  - Refactoring with confidence

- Errors and failures are different:

failures an explicit check failed (e.g. function returned 1 where 0 was expected)

errors unexpectedly aborted for some reason (usually segfault)

Known failures and skips

- Known failures specified in *samba4-knownfailures*
- Skipped testsuites specified in *samba3-knownfailures*

knownfailures

```
BASE-DELETE-deltest16
BASE-DELETE-deltest18
BASE-DELETE-deltest19
BASE-DELETE-deltest20
...
```

Environments

Supported Environments

none	Local, no server started
dc	Domain Controller
member	Domain Controller with Domain Member

- Each environment provides certain variables:
  - USERNAME, PASSWORD, DOMAIN, SERVER_IP
  - DC_USERNAME, DC_PASSWORD

Code coverage

- Gives some indication of completeness of tests
- Written out by programs compiled with *-ftest-coverage*
- Generated when running *make lcov*
- Now somewhere around 40%
- Available on host “tridge” on the buildfarm

A protocol for unit testing

- language independent
- standardized (somewhat)
- simple to generate and parse

The protocol

- start: *name*
- error: *name* [*reason*]
- failure: *name* [*reason*]
- skip: *name* [*reason*]

all other lines considered comments

Subunit example

```
test: list_empty
success: list_empty
test: share_create
skip: share_create [
torture/local/share.c:58: Not supported by backend
]
```

The old API

- Which API?
  - Sole requirement return code indication success or failure
  - Output not machine parseable
- Hard to machine-parse or run individual tests
- Lots of code repeated
- Not always as verbose as possible when failing

The new API

Macros

torture_fail(ctx, message);
torture_skip(ctx, reason);
torture_assert(ctx, expr, message);
torture_assert_ntstatus_equal(ctx, got, expected, message);
torture_assert_ntstatus_ok(ctx, got, message);
...

The new API

Macros

torture_fail(ctx, message);
torture_skip(ctx, reason);
torture_assert(ctx, expr, message);
torture_assert_ntstatus_equal(ctx, got, expected, message);
torture_assert_ntstatus_ok(ctx, got, message);
...

Output format

- Subunit by default
- As verbose as possible on failure
- Includes file and line information (**file**:**line**: error)

More Tests

- Tests added for existing libraries
  - lib/util
  - lib/replace
  - librpc/ndr
  - lib/registry

...

What?

1. Client opens connection to server
2. Client calls *spoolss_Openprinter*
3. Client calls *spoolss_RemoteFindFirstPrinterChangeNotifyEx*
4. Server opens connection to client
5. Server calls *spoolss_ReplyOpenPrinter*
6. Server calls for notifications

How?

1. register phony spoolss interface
2. start DCE/RPC server
3. start SMB server
4. run OpenPrinter
5. check to see if callback has been made
6. run ClosePrinter

Resources are automatically freed using talloc
Available as “RPC-SPOOLSS-NOTIFY” in Samba4’s smbtorture

Status

- Works against Samba 4
  - *make test TESTS=SPOOLSS-NOTIFY*
- Breaks against Samba 3 because of name resolution
  - Add NetBIOS name registration
- Tests for notifications trivial to add

3.2-perltest contains a backport of the new selftest code

- Uses *selftest.pl* from Samba 4 directly
- Brings a couple of nice features to Samba 3
  - *make test TESTS=SAMR*
  - *make testenv*
  - *make gdbtest*

  - Code coverage
  - Environments (domain member testing, . . .)
- Merged into Samba 3.2 this morning

Interoperability testing

- Test more combinations
- Run against Windows in buildfarm?
- Testing other combinations of Samba 3 and Samba 4
- Windows GUI testing
  - vmware, kvm?
  - Worked on by Brad Henry as part of SoC
  - on the buildfarm, maybe even make test?

Deployment testing

- All current tests run from within the development environment
- Installation is not necessarily valid or complete
- Installed headers should not break API compatibility
- Really often break e.g. openchange at the moment

Deployment testing

- All current tests run from within the development environment
- Installation is not necessarily valid or complete
- Installed headers should not break API compatibility
- Really often break e.g. openchange at the moment

- try compiling test apps with installed headers?
- store public function signature and check it doesn't change?

Benchmark tests in the buildfarm

- Make sure performance doesn't degrade?
- pretty hard, hardware dependent, etc
- how important?

More tests!

- Aim for 100% code coverage
- require test for bugfixes?
- distributed vcs - policy for requiring test successes?
- Upgrade testing
  - Upgrading from Samba3 should be tested
  - should be tested with real data created by samba 3