Online: Unit Testing

Michael Meeks

<michael.meeks@collabora.com>
mmeeks / irc.freenode.net
@mmeeks +mejmeeks

“In his great mercy he has given us new birth into a living hope through the resurrection of Jesus Christ from the dead” - 1 Peter 1:3
First - Terms & Architecture
Online: Unit Testing & Quality

Vital

- Allows us to create complexity & maintain it.
- Gives us confidence in our source releases

Hard

- Networking setup / latency etc. is tough.
- Code is split between processes – WSD, Forkit, Kit – with different permissions & capabilities
- Code is split between modules & abstracted behind LibreOfficeKit API

Harder: browsers

- Who wrote the front-end in Javascript?
- Browsers are quirky & different ...
- Visual / inspection of pixels is a horror ...

But: easier → Linux Only!
What tests do we have?

During Build - low dependency bits.

- TileCacheTests & WhiteBoxTests
  - Queue & preview priorities, tile combining, invalocation / page size pieces
  - Tokenizer, Regex matcher, Rectangle intersecter

New style

- Preforking, OAuth interactions, TileCache tweaks, Fuzzing plugin.
- Plus - the old-style tests (wrapped in a new-style test)

Old style

- Everything else:
  - Kit crash & recovery, Failed document load, Bad requests
  - load torture testing, save on disconnection(s), text selection
  - Copy/paste, password protection, slideshow, calc row/column
  - Graphic Selection, User Alerts, Repair-Document / Undo Conflicts ...

In-Browser - bit-rotted

- loleaflet/spec - runLoadTest.sh & other leaflet tests ...
## Old vs. New comparison

<table>
<thead>
<tr>
<th></th>
<th>Old Style</th>
<th>New Style</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Concurrency</em></td>
<td>Test + WSD + Kit*, Multi-thread</td>
<td>WSD + Kit*, Multi-thread</td>
</tr>
<tr>
<td><em>Debug-ability</em></td>
<td>Multi-process</td>
<td>Single Process</td>
</tr>
<tr>
<td><em>Logging</em></td>
<td>Multiple log streams</td>
<td>Console output</td>
</tr>
<tr>
<td><em>Performance</em></td>
<td>Lots of sleeps</td>
<td>Zero sleeps</td>
</tr>
<tr>
<td><em>Transparency</em></td>
<td>Acts like a normal client</td>
<td>Code injection &amp; hooks everywhere</td>
</tr>
<tr>
<td><em>Reliability</em></td>
<td>Opaque failures if WSD / test owns ports 9984 &amp;9985 Has to have SSL enabled</td>
<td>No dedicated ports required; Theoretically parallelizable; Certain of code run</td>
</tr>
<tr>
<td><em>Framework</em></td>
<td>CPPUNIT</td>
<td>Custom</td>
</tr>
</tbody>
</table>
Unit Testing tips.

- Ensure that SSL is enabled (for old-style)
- configure with –enable-debug – or some tests fail.
- Before running make check:
  
  $$ sudo \texttt{pkill} -9 \texttt{-f lool} \# \text{dung out existing wsd / kits.} $$
- Worth checking disk-space too: we warn and fail early.
- forkit has capabilities (cf. Root)
  - While these are dropped – you still can’t attach
    $$ sudo \texttt{gdb} \# \text{is your friend} $$
- sudo strace – but first patch:
  - if (geteuid() == 0)
  - throw std::runtime_error("Do not run as root ...")
- #define KIT_IN_PROCESS - Collapses whole architecture to one process.
- trace[@enable] and tools/Replay, tools/Stress ...
How New tests work
The flow

- test/Makefile.am
  
  `unit_prefork_la_SOURCES = UnitPrefork.cpp`
  
  `TESTS = unit-prefork.la` ...  
  
  - Add your test to TESTS

  $ make check
  
  - Watch the test fail: this is good ...

- test/run_unit.sh -test-name `unit-prefork.la`

  - generated from run_unit.sh.in by configure / config.status
  
  - runs tests & logs to stderr

- test/run_unit.sh --help
Writing your test
Bare bones of a new unit-test:

- Magic entry point:
  ```c
  UnitBase *unit_create_wsd(void) // Called in WSD
  {
    return new UnitFuzz();
  }
  UnitBase *unit_create_kit(void) // Called in Kit
  {
    return new UnitKitFuzz();
  }
  ```

- Sub-class common/Unit.hpp
  - UnitWSD & UnitKit

- Sample hooks – easy to add more:
  ```c
  /// Main-loop reached, time for testing
  virtual void invokeTest() {}
  /// When admin notify message is sent
  virtual void onAdminNotifyMessage(const std::string& /* message */) {
    ...
    exitTest(TestResult::OK | Failed | TimedOut); ...
  }
  ```
Bare bones of a probe ...

- Add it to Unit.hpp – UnitBase / UnitWSD / UnitKit
  - Filter pattern allows us to inject changes to the control flow:
    ```
    /// Trap and filter alerting all users
    virtual bool filterAlertAllusers(const std::string & /* msg */) {
      return false;
    }
    ```

- Invoke the filter and act on its output where you like:

  ```
  void DocumentBroker::alertAllUsers(const std::string& msg) {
    if (UnitWSD::get().filterAlertAllUsers(msg))
      return;
  }
  ```
WSD: What hooks do we have?

I/O bits:

- **handleHttpRequest**(const Poco::Net::HTTPRequest& req,
  
  std::shared_ptr<StreamSocket>& socket)

  - Filter any incoming HTTP request

- **filterHandleRequest**(TestRequest type (Prisoner or Client),
  
  SocketDisposition &disposition, WebSocketHandler &handler)

  - Allow filtering of raw WebSocket protocol inputs

- **filterSessionInput**(Session *, const char *buffer, length, std::unique_ptr<std::vector<char>> &replace)

  - Filter or mutate parsed data from the WebSocket

Misc / Warnings

- **filterCheckDiskSpace**, **filterAlertAllUsers**

- **configure** → allow clobbering any configuration items

- **onChildConnected**

TileCache → **onTileCacheHit / Miss / Subscribe**
What other hooks do we have?

Admin

- onAdminNotifyMessage / onAdminQueryMessage
  - Filter / test incoming / outgoing Admin Console traffic.

Kit bits

- FilterKitMessage – allows hooking Kit specific messages via old LOOLWebSocket
  - launchedKit – hook just after we fork to initialize the child.

ForKit

- InvokeForKitTest – run only in the forkit process
- launchedKit(int pid) – when we’ve launched a kit
Summary

- Unit testing is vital
- You should write tests
- There are several ways to do it
- Use the ‘new’ way if you can
  - Add probes / instrumentation to the code as you go to test.
  - More (reliable) automated tests are always appreciated
- Poke me – if you need help writing a test.

Oh, that my words were recorded, that they were written on a scroll, that they were inscribed with an iron tool on lead, or engraved in rock for ever! I know that my Redeemer lives, and that in the end he will stand upon the earth. And though this body has been destroyed yet in my flesh I will see God, I myself will see him, with my own eyes - I and not another. How my heart yearns within me. - Job 19: 23-27