# Landslide: Systematic Testing in 15-410 User Guide and Survey

Ben Blum

## 1 Introduction

You should have received a document titled *User Study Information Sheet* that describes your rights as a study participant and our confidentiality procedures. If you are missing this, please contact me immediately. My email address is `bblum@cs.cmu.edu` and my phone number is 412-304-4294; email is preferred.

Likewise, please make sure you have read and understand the lecture slides on systematic testing, which are available on the 15-410 course website. If you have any questions about the research background I would be happy to answer them.

## 2 Instructions

The very first thing you should do is **READ THIS DOCUMENT ENTIRELY FIRST!** We need you to print out several survey sheets before you dive in.

We encourage you to use Landslide on the GHC cluster or UNIX pool machines (`ghcXX.ghc.andrew.cmu.edu` or `unix.andrew.cmu.edu`). Landslide has not been tested with other configurations and will probably fail mysteriously on your personal laptop, and we cannot promise to be of any help.

1. Clone the repository – `git clone https://github.com/bblum/landslide.git` – this will be your workspace.
2. Run `./p2-setup.sh PATH`, where `PATH` is the absolute path to your p2 implementation.
    - This command will import your p2 into `./pebsim/p2-basecode/` and then attempt to build it.
    - `PATH` should be the path with `410kern`, `410user`, `user/libthread`, etc as subdirectories, not the `user/libthread` directory itself.
3. Next, make sure you have read this document entirely. At this point you should have several survey sheets printed out, ready to write on – see the Survey section below. You should also have some means of timing yourself.
4. Run `./landslide OPTIONS` to run tests with Landslide. **Fill out a fresh survey sheet each time you do this.** Repeat until satisfied.
    - Use `./landslide -h` to view command line options. The most important ones are `-t`, max testing time, and `-p`, which test program to run.
    - Landslide's bug reports will show up as HTML files. If you are using a cluster machine in person, you can view them by pointing a browser to `file:///PATH/TO/landslide/FILENAME.html`. [1] [2]

[1] Another option: move it to `~/www` in your AFS, and view it at `http://www.contrib.andrew.cmu.edu/~YOUR_ANDREWID/FILENAME.html`. Note the `www` in the URL is mandatory; note also that this will allow the world to see it, so be careful!

[2] If you have your own custom test case, you can put it in `pebsim/p2-basecode/user/progs` and edit `pebsim/p2-basecode/config.mk` to build it, then run `./p2-setup.sh` again.

- The available test programs are `thr_exit_join` (the default), `broadcast_test`, `paraguay`, and `rwlock_downgrade_read_test`.
- The suggested sequence of test configurations to run is as follows. You can, of course, change the time limits or test programs as you see fit. For the longer-running configurations, it's recommended to run them inside of a `screen` session.
  - `./landslide -p thr_exit_join -t 30m`
  - `./landslide -p broadcast_test -t 30m`
  - `./landslide -p paraguay -t 30m`
  - `./landslide -p rwlock_downgrade_read_test -t 30m`
  - `./landslide -p thr_exit_join -t 3h`
  - `./landslide -p broadcast_test -t 3h`
  - `./landslide -p paraguay -t 3h`
  - `./landslide -p rwlock_downgrade_read_test -t 3h`
  - `./landslide -p thr_exit_join -t 12h`
  - `./landslide -p broadcast_test -t 12h`
  - `./landslide -p paraguay -t 12h`
  - `./landslide -p rwlock_downgrade_read_test -t 12h`

**What to do when something goes wrong:** If you find a Landslide bug, such as an assertion/crash, or such as a bug report that you think is actually a bug in Landslide rather than a bug in your P2, or such as it getting stuck and not exiting in the given time limit, please create a tarball of your workspace, just as you would to report a reference kernel bug, but send it to Ben (`bblum@cs.cmu.edu`) instead of 410 staff. For any other technical issues getting it to work, or if you have difficulty understanding a bug report, send email to Ben.

However, we *cannot* provide help on how to understand *why* your bug happened or how to fix it. The point of the study is that Landslide's automation plus your own brain should be enough! On the other hand, if you have a design question while deciding between potential ways to fix a bug, send it to 410 course staff as usual.

## 3 Survey

### 3.1 Overall Survey Questions

**Print this page out once.**

1. How much time did you spend directly using Landslide? This includes configuring test parameters and interpreting debug output. It does not include time spent running Landslide in the background, e.g. overnight.

2. What test configurations did you use that DIDN'T find bugs? For example, `./landslide -t 15m -p thr_exit_join`

## 3.2 Per-Bug Survey Questions

**PRINT OUT THIS PAGE MULTIPLE TIMES.**
**Each time you find a new bug with Landslide, fill this out again.**

1. What test configuration did you use? For example, `./landslide -t 15m -p thr_exit_join`

2. Summarize the bug.

3. How IMPORTANT was this bug? Rate on a scale from 1 to 10, where:
   - 1 = Trivial; an obvious "oops" that does not reflect any key lessons learned during the project. I would expect to not lose any points for this bug.
   - 4 = Minor; a logic problem not related to threads/synchronization. I would expect to lose 1 or 2 points at most for this bug.
   - 7 = Major; a concurrency-related bug related to subject matter discussed in lecture.
   - 10 = Severe; this bug reflects a fundamental lesson about concurrency that we learned during the project. I would expect to receive "see course staff" in my ink if I turned in code with this bug!

4. How much EFFORT/TIME did this bug require to diagnose and fix? Rate on a scale from 1 to 10, where:
   - 1 = Very little; we knew immediately what was wrong and needed to change only 1 or 2 lines of code.
   - 5 = Some; the bug took considerable thought to fix, and required nontrivial code changes.
   - 10 = Lots; we worked for several hours to fix the bug and/or required a major refactoring of our code.

5. How likely would a CONVENTIONAL STRESS TEST be able to find the bug (i.e., running the program repeatedly, not using Landslide)? Rate on a scale from 1 to 10, where:
   - 1 = No hope. The window for a random timer interrupt to expose the bug is so small that a conventional stress test could easily run continuously overnight without encountering it.
   - 5 = Moderate chance. Most random timer interrupts would not expose the bug, but the window is more than just a few instructions.
   - 10 = Very likely. A random timer interrupt during the test would expose the bug more often than not.