Appendix C

User Guide

The following document provides instructions on how to compile and run the diagnostic tool. It then explains how to use the tool in more detail.

C.1 Running the diagnostic tool

The diagnostic tool can be run from the command line on Mac OS X or Linux by navigating to the `fdt` directory and running the following command:

```
./fdt
```

This will open the GUI. An application bundle for Mac OS X is also included which allows the tool to be run by double-clicking "FUSE Diagnostic Tool" in the `fdt` directory.

C.2 Running the test data logger

The command line tool for logging call sequences is run using the following command:

```
./fdt --logger [-a] [FUSE Binary] [FUSE Arguments]
```

The FUSE filesystem binary and its arguments should be passed to the logger, which will mount the filesystem and allow calls to be logged. The `-a` parameter should be passed when it is necessary to start logging calls as soon as the filesystem mounts, as calls like `init` will be made straight away. The logger displays available options as it runs, allowing capture sessions to be started and stopped as required. Once a capture session is terminated, the tool will prompt for metadata including a high-level description of the action performed during the session. The tool allows data to be exported when pressing "q" to terminate.

C.3 Compilation

It may be necessary to compile parts of the tool to run it on your platform. These parts include the main frontend of the tool, the modified `libfuse` library (if running Linux) and the modified `osxfuse` library (if running Mac OS X).

The main frontend is compiled by navigating to the `fdt` directory and typing `make`. This depends on the `gtk+-2.0` development libraries which the makefile will locate using `pkg-config`. On Mac OS X, the makefile will invoke the `jhbuild` environment to find GTK+, which can be installed by following the GTK-OSX build instructions: `http://www.gtk.org/download/macos.php`.

`libfuse` is compiled by navigating to the `libfuse` directory and typing `make`. `osxfuse` is compiled by navigating to the `osxfuse` directory and typing `sudo ./build -t dist`. Although the build process on Mac OS X requires root access, it does not permanently install the modified library to the system.

C.4 Using the diagnostic tool

A FUSE module can be selected by clicking the "Browse..." button next to the "FUSE Binary" field. This will open a window to find the binary for the FUSE module. Alternatively, the command or path for the FUSE module can be typed into the field. Most FUSE modules require a number of arguments to be passed on mount, which can be specified in the field underneath the binary field. At a minimum, a mountpoint path must be specified as this is required for all FUSE modules.

C.4.1 Using the wizard

1. Make sure a FUSE module is selected and all arguments are set (described above).
2. Select the Wizard tab.
3. Click Start. This will begin running tests against the provided FUSE module.
4. The table on the left lists the functions that have been tested, indicating whether or not each function is implemented correctly (or at all). If a function is incomplete then instructions will be displayed to the right, explaining how to implement the functionality and providing documentation on the function.
5. If a function fails a test but that functionality is not required for the FUSE module, press the "Skip" button to run the wizard again ignoring that test. It is possible to skip multiple tests and these preferences will persist until the application is closed.

C.4.2 Using the test suite

1. Make sure a FUSE module is selected and all arguments are set.
2. Select the Test Suite tab.
3. Click the "Browse" button next to the "Test Data" field to select the test data file. This is the JSON-formatted collection of call sequences exported by the logger tool.
4. Click Start. This will begin replaying the calls against the provided FUSE module, comparing the results of each call against its expected behaviour. It will also compare all visible filesystem state, including the existence of files and their metadata.
5. Test results are displayed in the table, and broken down into Groups, Sequences and Calls. If a call fails then this fails the sequence, which fails the group. Expanding a failed call will reveal more information about the failure, including the parameters and return value, the expected behaviour, and any error that occurred or should have occurred.

C.4.3 Using the debugger

1. Make sure a FUSE module is selected and all arguments are set. The mountpoint must be valid otherwise the filesystem will not mount.
2. Select the Debugger tab.
3. Click Start. This will mount the filesystem and begin displaying API calls in the table.
4. Click Advance to allow the `init` call to execute and return. Ticking the Auto-advance checkbox will advance calls automatically, including any currently held call.
5. Click the arrow next to an "invoke" event to display the values of the parameters passed to that function call.
6. Click the arrow next to a "return" event to display the values of parameters that the function may have modified.