What Changed

Sometimes files change in the file-system without the user knowing. It could cause file-corruption, and errors that could’ve been potentially avoided if a modified/missing file was reported.

Problem:

* Files are changed without the user noticing.

Solution:

* A system to monitor file changes, and report them to the user.

Requirements specifications:

* Compare old file-state with the new one, and generate a report containing the differences.
* Keep track of changes in real-time.
* It should be light to allow for real-time operation without loading the system.
* The reports/interface should be user-friendly.
* A free-license database should be used.
* Operating environment:
  + The system is tested on Window 7 x64.
  + It requires JRE 7.
* The user manual will be in a .doc format.

Features:

* Select files to be kept track of.
* Save file-state in a database when snapshotted.
* Generate a report for the current file-state.
* Generate a report of the difference between the last state and the current state.
* Show a summary of changed files.
* Show a detailed report of all files selected.
* Monitor files in real-time and pop a notification when something happens.

GUI:

* The interface is divided into 2 sections vertically and 2 horizontally.
* Horizontally, the first section is for selecting files, the second is for showing what files were selected.
* Vertically, the first section is as above, the second is for real-time logging of changes.
* Checkbox selections.
* A button for refreshing the tree of folders/files.
* A button to change the root drives.
* A button to remove files from selection.
* A button to exit the application.
* A button to take a file-state snapshot.
* A button to show a summary report.
* Inside there’s a button for showing a full report, which is in tabular form, with files, their sizes, modification date, and full path.
* The summary report is for showing just the names of files changed.

‘Generate Report’ test case:

1. When the program starts, it should present an ‘open-file’ dialogue, showing the list of root entries in the file-system.
2. Deliberately choose the wrong entries, and then change the entries after the main-window appears.
3. Make sure the tree mirrors the actual file-system.
4. Expand the tree to a level greater than 2, and make sure it still mirrors the file-system.
5. Choose random files from random levels greater than 2, then make sure the files are accurately listed on the right-side, in-order.
6. Remove a file from the list and make sure its tick is removed from the left-side.
7. Take a snapshot, and then make sure its time is correct.
8. Generate a report, but a message should say everything is intact.
9. Change any file listed, and then re-generate a report; a summary should appear listing the files changed, each under the appropriate type of change. The number of files should be reflected on top.
10. The full-report should show the file-state at snapshot-time, and the current file-state with difference in size represented as a percentage.
11. Upon restarting the program, the file-list should be restored, the snapshot time, and the file-state at that snapshot time.
12. To make sure, change a file, and then check the full-report, it should reflect a time before the snapshot time on the right side.

‘Real-time monitoring’ test case:

1. Perform steps 1-6 in the previous test-case.
2. Enable monitoring, and then change a file from the list. A pop-up should appear within 1 second reflecting the change-type, and the file-name. In addition, a log entry and the file-counter should reflect such change.
3. If more than one file is changed within 3 seconds, the pop-ups should appear above each other; without overlapping.
4. If a file is added or removed from the list, and then monitoring should be automatically disabled.
5. Disable monitoring, and then change a file, it shouldn’t pop a notification.