Item 57
git_comments:

1. LUCENE-6279: don't rely solely on existence of the marker file; also require that we see the marker file in our si.files(), which means we did previously at least attempt to write it:
2. Also verify the marker file exists and has the proper header:
3. We intentionally double-write the upgrade marker file:
4. LUCENE-6279
5. Causes FNFE on _0.si during check index before the fix:
6. Create errant leftover file, after opening IW but before closing IW:

git_commits:

1. **summary:** LUCENE-6279: don't let a leftover _N_upgraded.si file corrupt a 3.x index on first kiss from 4.x
   message: LUCENE-6279: don't let a leftover _N_upgraded.si file corrupt a 3.x index on first kiss from 4.x git-svn-id: https://svn.apache.org/repos/asf/lucene/dev/branches/lucene_solr_4_10@1662024 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
   description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.
2. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
   description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.
3. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
   description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.
4. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
   description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.
   label: code-design
5. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
   description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.
   label: code-design
6. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
   description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.
7. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
   description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.
8. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
   description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this

process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.
9. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
   description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.
10. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
    description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.
11. **summary:** 3.x -> 4.x .si upgrade should not be tricked by leftover upgrade marker file
    description: Today when you do the first IW.commit to a 3.x index from Lucene 4.x, we go through a per-segment upgrade process when writing the next segments_N file, writing .si files for each segment if we didn't already do so. However, this process can be fooled by a leftover _N_upgraded.si file, in case the app above Lucene wasn't careful and reused a directory that had leftover files... I think we can make this more robust.

jira_issues_comments:

1. Test case showing the issue (applies to 4.10.x); the test fails with this: {noformat} java.io.FileNotFoundException: _0.si in dir=RAMDirectory@3833bc67 lockFactory=org.apache.lucene.store.SingleInstanceLockFactory@69b2486e at __randomizedtesting.SeedInfo.seed([1598963DC9C89C28:2D64E0AACDD85FF7]:0) at org.apache.lucene.store.MockDirectoryWrapper.openInput(MockDirectoryWrapper.java:603) at org.apache.lucene.codecs.lucene3x.Lucene3xSegmentInfoReader.read(Lucene3xSegmentInfoReader.java:106) at org.apache.lucene.index.SegmentInfos.read(SegmentInfos.java:358) at org.apache.lucene.index.SegmentInfos$1.doBody(SegmentInfos.java:454) at org.apache.lucene.index.SegmentInfos$FindSegmentsFile.run(SegmentInfos.java:906) at org.apache.lucene.index.SegmentInfos$FindSegmentsFile.run(SegmentInfos.java:752) at org.apache.lucene.index.SegmentInfos.read(SegmentInfos.java:457) at org.apache.lucene.index.CheckIndex.checkIndex(CheckIndex.java:414) at org.apache.lucene.util.TestUtil.checkIndex(TestUtil.java:207) at org.apache.lucene.store.MockDirectoryWrapper.close(MockDirectoryWrapper.java:724) at org.apache.lucene.index.TestBackwardsCompatibility3x.testLeftoverUpgradedFile(TestBackwardsCompatibility3x.java:1038) {noformat}
2. I've thought about a couple ways to fix this. We could fix SIS to record on read that it was pre-4.0, and then on write it upgrades any 3.x segments. This is kinda a big change, though, and I don't get why we do it today on every commit ... is there any way for a 3.x segment to sneak in to a 4.x SIS un-upgraded? IW.addIndexes(Dir[]) writes upgrades .si ... A less scary change would be to just check the SI.files() to see whether the .si file is in there; if it is, it was already upgraded. Then we could remove the marker file entirely. Yet another maybe even less scary option would be to keep writing the marker file, but check if it's already in SI.files() (instead of trying to open it from the filesystem) to see whether we already upgraded...
3. Hmm this test is obviously abusive?
4. **body:** bq. Hmm this test is obviously abusive? It is abusive: it can happen if the caller restores an index into a "dirty" directory already containing these marker files, which is obviously not a good idea. It does make me nervous to change this back compat logic, but then again I don't like that it currently relies on a file existence check, when it already has its own more reliable internal state making it clear whether the upgrade was done.
   label: code-design
5. **body:** The only real viable solution is for 3.x si to be a per-commit file, or some very invasive change. I don't think we should change this stuff in this way, in a bugfix release, because of abuse cases. Seems like it would be easier to just fix the abusers.
   label: code-design
6. OK I agree...
7. OK I think I found a nice low-risk change, so we don't rely on File.exists to decide when to write the .si for a 3.x segment: I just inserted an additional check, that we see this marker file name in our si.files() already, and if it's not there, we always write it.
8. +1
9. Commit 1662024 from [~mikemccand] in branch 'dev/branches/lucene_solr_4_10' [https://svn.apache.org/r1662024] LUCENE-6279: don't let a leftover _N_upgraded.si file corrupt a 3.x index on first kiss from 4.x