Item 63
git_comments:

1. if we couldn't resolve the lock file, it might be because we couldn't create it. so append any exception from createFile as a suppressed exception, in case its useful
2. create a mock filesystem that will throw exc on creating test.lock
3. we should get an IOException (typically NoSuchFileException but no guarantee) with our fake AccessDenied added as suppressed.
4. * MockFileSystem that throws AccessDeniedException on creating test.lock

git_commits:

1. **summary:** LUCENE-7959: Improve NativeFSLockFactory's NoSuchFileException
   message: LUCENE-7959: Improve NativeFSLockFactory's NoSuchFileException
   label: code-design

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Throw more helpful error messages from failures in obtainFSLock, at least in NativeFSLockFactory
   description: This is one of those small changes that would save a lot of pain for end users. Let's say that there's a permissions issue with the index dir. The user will get "NoSuchFileException: write.lock" but not understand why the lock file is missing (its because it could not be created). This is an unhelpful/misleading error message (recent user's list discussion about this). The problem is that when we try to create the lock file we swallow _all_ exceptions, not just the one we don't care about (Thanks [~elyograg] for pointing this out). {{ try { Files.createFile(lockFile); } catch (IOException ignore) { // we must create the file to have a truly canonical path. // if it's already created, we don't care. if it cant be created, it will fail below. } }} It fails later with NoSuchFileException, which does not provide enough information.

jira_issues_comments:

1. I want to track this so I assigned it to myself but would be _very_ happy if someone else took it.
2. The code is correct: its not possible to catch anything more specific because of how the api works. Anything other than IOException is "optional specific exception" and any fs provider may or may not implement them, including FileAlreadyExistsException, AccessDeniedException, etc. See: https://docs.oracle.com/javase/7/docs/api/java/nio/file/Files.html#createFile(java.nio.file.Path,%20java.nio.file.attribute.FileAttribute...)
3. by the way, if you have a permissions issue where you don't have access to a file, you will get that access denied exception exactly from the FileChannel.open. So we aren't masking any exceptions here at all: We throw "Lock held by this virtual machine" if there is already an entry in LOCK_HELD map, and we throw "Lock held by another program" if tryLock() returns null. I think we can safely close the issue.
4. **body:** This is what I think should be there, including some comments: {code} // If the lockfile already exists, we're going to do nothing. // If there are problems with that lockfile, they will be caught later. // If we *do* create the file here, exceptions will propagate upward. if (Files.notExists(lockFile)) { Files.createFile(lockFile); } {code} If the file already exists, it won't try to create it, and the later code can blow up like it already does if the lock fails. If the file doesn't exist, then creation is attempted. If that creation fails, and Java can create a reasonable exception explaining WHY it failed, then the calling code will have that information. The problem I see with the current code is that when lockfile creation fails, the *reason* for the failure is completely lost. An exception is thrown later, but it's an almost useless "NoSuchFileException", which is covered by the method's signature (that includes IOException). It would be better to have the file creation call throw an appropriate IOException, or one derived from it. Then the caller will know that permissions are wrong, or they're out of disk space, or some other problem.
   label: code-design
5. We can't do sneaky stuff like what you propose because it would be full of race conditions.
6. {quote} An exception is thrown later, but it's an almost useless "NoSuchFileException" {quote} OK, this is the key: this happens in the call to .toRealPath(). So I think there is an easy safe fix, when we try to createFile(), save the exception we caught, and if toRealPath() then fails with IOE, addSuppressed(savedException) and rethrow it. It will never hurt anything.
7. **body:** Right, could we at least extract the message from whatever exception is thrown in that block where we swallow everything and add it to the failure message, assuming we can't obtain the lock? That would avoid race conditions & etc. but still provide (perhaps) a more useful error message? Neither of the two error messages reported reflect the reality of a permissions issue found by this particular case. Well, just saw your other reply which looks better
   label: code-design
8. Here is just demonstrating my idea, i think its pretty conservative/safe: it just adds additional context via addSuppressed(), doesn't try to interpret anything about the IOExceptions at play. But I want to think about it some more...
9. [~rcmuir] Feel absolutely free to assign this JIRA to yourself. Or I'll be happy to apply/test/precommit/whatever.....
10. **body:** Erick well i still think there is confusion in the issue description at least. To clarify: If you get lockobtainfailedexc, its a bug in your code always. This is about the case where no lock file exists yet (no index yet) and we are unable to create it due to perms or disk space or whatever. Today you get NoSuchFileException or some other IOException in this case, it may not be enough to help you. So the suppressed exc can be useful then, maybe. It could be that toRealPath fails for some other reason, in which case hopefully the additional stuff is not confusing. But presumably its the most likely situation.

label: code-design

11. **body:** +1 to the patch, except some small typos: {noformat} i ncase its useful --> in case it's useful {noformat}
    label: documentation
12. **body:** bq: i still think there is confusion in the issue description at least feel free to change as you see fit. All I really are about is that more clues about what's _really_ wrong if we can safely provide them show up in the logs. bq: If you get lockobtainfailedexc, its a bug in your code always. agreed. nit: in this case not code, your setup. But the point is still well taken, you have a problem you've created for yourself and you have to fix it. This is about providing clues to help them fix it. bq: So the suppressed exc can be useful then, maybe Right, not looking for too much hand-holding here, just anything that could help the user (or me if I'm supporting them) figure out what the underlying issue is. The errors currently returned are misleading in this case; there is no lock held at all by a VM or external process. So any safe thing we can do that has a _chance_ of helping pinpoint the problem is an improvement.
    label: code-design
13. Hi, The patch looks functionally correct, adding a supressed ex is fine. If we supress some exception, it's always a good idea to provide it on an real error condition. This what addSuppressed() is made for. Adding a test may work, but only with with some proper setup of virtual file systems. Do we have anything at hand that may emulate an IOEx when creting a file?
14. {quote} The errors currently returned are misleading in this case; there is no lock held at all by a VM or external process. So any safe thing we can do that has a chance of helping pinpoint the problem is an improvement. {quote} Erick, I'm trying to reiterate here, its super-important to understand, that if you see "Lock held by this virtual machine" or "Lock held by another program", it is absolutely a bug in your code. Those errors have nothing to do with this issue, you simply have to fix your code. This issue is only about improving the "NoSuchFileException" case, that is it.
15. **body:** {quote} bq: If you get lockobtainfailedexc, its a bug in your code always. agreed. nit: in this case not code, your setup. But the point is still well taken, you have a problem you've created for yourself and you have to fix it. This is about providing clues to help them fix it. {quote} I updated the description, but i also must correct this "nit": If you get {{org.apache.lucene.store.LockObtainFailedException: Lock held by this virtual machine}}, you need to stop reading this JIRA issue and go fix that bug in solr! That happens because you try to open two indexwriters in the same JVM over the same index, and for no other reason. Its not related to this JIRA or changed by this patch in any way. On the other hand, if you get {{org.apache.lucene.store.LockObtainFailedException: Lock held by another program}}, then it happens because *two different operating system processes* each try to open an indexwriter over the same index, and no other reason. In that case, you may want to ask the user how many solr processes are running on the machine, etc. Maybe they managed to screw up and start it twice somehow. The two error messages are intentionally different for this reason, so that its easier to tell when there is a software bug that needs to be fixed, versus an operational screw-up or what have you.
    label: code-design
16. **body:** updated patch with a test like [~thetaphi] suggests.
    label: test
17. **body:** OK, maybe I get it now. Credit my slow uptake to slightly different versions from the original problem statement and not looking carefully enough at the code. Playing around on my Mac, I get what _I_ expected at least: Caused by: java.nio.file.AccessDeniedException: which is pretty clear. The original problem reported: Caused by: java.nio.file.NoSuchFileException: and the OP said "it was a permissions issue". So I jumped to conclusions. The implementation on the original statement throws a different error..... Or I didn't reproduce the right conditions. Or.... Neither of them ever got to the two hand-crafted error messages about a lock being held by another process or virtual machine, I jumped over the toRealPath() line and focused on the lines above them, which with an un-careful reading looked like it swallowed all exceptions. Which actually it does but it doesn't matter since the toRealPath line is where the error is thrown from. Just like some guy named Muir said.... Siiiiggggghhhhh. So anything we can add safely I'm all for. Sorry for the noise...
    label: code-design
18. Test looks good! Thanks Robert!
19. Commit 0283d3e28a6dea083060cb6932bca82a9b8fefb2 in lucene-solr's branch refs/heads/master from [~rcmuir] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=0283d3e] LUCENE-7959: Improve NativeFSLockFactory's NoSuchFileException
20. Commit bcdac5fef3ac0a8de074ca8308206e39d81e439f in lucene-solr's branch refs/heads/branch_7x from [~rcmuir] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=bcdac5f] LUCENE-7959: Improve NativeFSLockFactory's NoSuchFileException
21. Thanks [~erickerickson] [~elyograg]
22. Bulk close after 7.1.0 release