## Behavior of git checkout

Checking out an earlier commit will change the state of at least one file.

This is sometimes true. Git doesn't allow you to save a new commit if no files have been updated, so you might think this is always true. However, it's possible to do the following:

* Save a commit (call this commit 1).
* Update some files and save another commit (call this commit 2).
* Change all the files back to their state during commit 1, then save again (call this commit 3).

This sometimes happens if commit 2 contained a bug, and it's important to fix the bug quickly. The easiest thing to do might be to remove all the changes introduced by commit 2 to fix the bug, then figure out how to safely reintroduce the changes later.

At this point, commit 3 is the latest commit, so if you checkout commit 1, none of the files will be changed.

Checking out an earlier commit will change the state of more than one file.

Checking out an earlier commit will change the state of every file in the repository.

Both of these are sometimes true. Since each commit tracks the state of all files in the repository, it is possible that checking out an earlier commit will change the state of multiple files, or even all the files in the repository. However, it is possible to save a new commit after changing only one file, so it is possible only one file will change.

After checking out a commit, the state of all the files in the repository will be from the same point in time.

This is always true. A commit saves a snapshot of all files in the repository at the time the commit was made, so checking out an earlier commit will result in all the files being reverted to their state at the time the commit was made. That is, the files will be in a consistent state.