Directory watching API for repo rules

- **Authors:** wyv@bazel.build (Xudong Yang)
- **Status:** Implemented (7.1.0)
- **Reviewers:** fabian@meumertzhe.im (Fabian Meumertzheim), lberki@google.com (Lukács T. Berki), pcloudy@google.com (Yun Peng)
- **Created:** 2023-05-02
- **Updated:** 2023-05-04

Please read Bazel Code of Conduct before commenting.

Background

- Module extensions can invoke repo rules to define repos, but it's a long-standing limitation that they cannot invoke native repo rules (as in, those implemented in Java instead of Starlark) (#15412).
  - Bazel currently has 5 native repo rules.
    1. `local_repository`
    2. `new_local_repository`
    3. `local_config_platform`
    4. `android_sdk_repository`
    5. `android_ndk_repository`
  - Thus we decided to Starlarkify those 5 native repo rules instead (#18285).
  - Out of the five rules, 1 and 3 are trivially Starlarkifiable. The other three are currently not, as they get a `DirectoryListingValue` out of Skyframe one way or another, and that's not possible with the current repo rule API.
- Additionally, the fact that `path.readdir` doesn't cause the repo to be refetched when the contents of the directory in question change has been reported as a bug (#14200). There is currently no available API to resolve this.

Proposed solution

Augment `path.readdir`

- We augment `path.readdir` to additionally register a Skyframe dependency on the corresponding `DirectoryListingValue`.

    - That is, a call to `path.readdir` will cause Bazel to "watch" that directory, and refetch the repo whenever that directory's contents (as in direct children) have changed.
    - This is technically not a breaking change, as the only consequence is that we trigger refetches on repos potentially more often. It also triggers more repo rule restarts, which is more dangerous (due to #10515).
- We also write the hash of the returned `Dirents` as an entry into the marker file of the repo.
    - For those not familiar, the marker file of a repo is used to check whether a refetch is needed. Conceptually, it's the "repo cache key".
    - The entry will have the key "`DIR:`" followed by the path to the directory.
- Note that we need to consciously *not* watch any directory located within the current working directory, i.e. the output directory of the repo we're constructing.
    - "Watching" such a subdirectory is not only useless (as we *just* created this subdirectory at some earlier point during the repo rule evaluation), but also results in a Skyframe dependency cycle.

Augment `ctx.read` for symmetry

- Today, we "watch" the contents of a file to trigger refetches, if a `Label` to said file was ever converted to a `path` by any means during repo rule evaluation.
    - The conversion could happen via an explicit `rctx.path(some_label)` call, or when a label is passed to any path-taking method (for example, `rctx.read(some_label)`).
    - The reason why we only do this for paths converted from `Label`s is unclear. This functionality was introduced in 44847d8.
- This means that even if we read the contents of a file during repo rule evaluation, we don't trigger a refetch when that file changes unless that file was referred to using a `Label`.
    - For example, `rctx.read("/etc/myconfig.rc")` doesn't watch `/etc/myconfig.rc`, though `rctx.read(Label("//:myconfig.rc"))` watches `//:myconfig.rc`.
- Since we're making `path.readdir` "watch" the contents of the directory (no matter whether it is addressable by a label), it would make sense to have any method relying on the contents of a file "watch" the contents of that file.
    - In other words, such a method would register a `FileValue` dependency, and cause the hash of the file in question to be written to the marker file.
    - Impacted methods include:
        - `rctx.read`
        - `rctx.symlink` (for the symlink target only)
        - `rctx.template` (for the template file only)
        - `path.exists`
        - `path.realpath`
    - Notably, simply creating a `path` object no longer necessarily causes Bazel to watch the file the path points to.

        - Except that, in the interest of backwards compatibility, we need to retain the behavior of watching a file whenever a `Label` is converted to a `path`, in part because it's used as a workaround for avoiding repository rule restarts.
- Since a watched file is no longer necessarily addressable by a label, we need to change the marker file format for `FILE:` entries.
    - The key of such entries used to be `FILE:` followed by an absolute label.
    - We can simply change it to be `FILE:` followed by an absolute path.
        - Since the marker file is never shared between different machines, this is safe.
    - We'll increment the marker file version for this change (effectively invalidating all fetched repos, forcing a refetch of everything once the user upgrades their Bazel to this version).
- Similar to the concern with `path.readdir`, we need to make sure *not* to watch any file located within the output directory of the repo we're constructing.
    - This didn't use to be a problem, since it was effectively impossible (or, at least, crazy) to acquire a label pointing into the repo under construction.

Alternatives considered

An explicit `rctx.watch` API

- Instead of implicitly registering a Skyframe dependency on every `path.readdir` call, we could have an explicit `rctx.watch_dir()` API.
    - Similarly, instead of `rctx.read()` & co registering a `FileValue` dependency, we could have an explicit `rctx.watch()` API.
- Pros:
    - We avoid accidental refetches of the repo if we need to list the children of a directory (or read the contents of a file) but for some reason *do not* want to refetch if that directory / file is changed.
    - As any Skyframe dependency registration triggers a restart of the repo rule impl function, users could consciously group calls to `watch`/`watch_dir` at the top of the impl function to make restarts less expensive.
        - This is similar to the trick we have today, where we ask authors of long repo rules to group their `rctx.path(Label(...))` calls at the top of the impl function.
    - An explicit API allows us to watch a file/directory without actually reading the file or listing the directory contents in Starlark. This might be the case if, for example, we're actually reading the file in another program run by `rctx.execute`.
- Cons:
    - It's unclear that anyone would ever want to avoid refetches if they call `path.readdir` and the directory contents change.

        - Having these separate APIs goes against the "sensible default" principle. Users will likely forget to call `watch` when necessary.

A `watch` boolean parameter to the methods above

- We could add a `watch` boolean parameter to the methods mentioned above (including `path.readdir`, `rctx.read`, `rctx.symlink`, etc.) to specify whether changes should cause a refetch.
- Depending on the default value of `watch`, this has similar pros and cons to either the proposed solution (if `watch` defaults to `True`) or the explicit API alternative (if `watch` defaults to `False`).
- Extra pros compared to the proposed solution:
    - Gives the user more control, while retaining a sensible default.
- Extra cons compared to the proposed solution:
    - More API surface, meaning more cognitive burden and chance to foot-shoot.

Document History

Date	Description
2023-05-04	First proposal

Archive

Which API do we attach this to?

- We could always introduce a new method like `repository_ctx.list_directory(some_path)`.
- But we already have `Path#readdir`. It currently calls `readdir()` on the given directory path, and returns the names of the direct children (sans their file types etc).
    - The problem is that it doesn't currently register a Skyframe dependency on anything, so if that directory changes, the repo rule is not rerun.
    - But do we just change it to use a `DirectoryListingValue`? This opens up more questions.
        - Is that backwards compatible? Mostly… since running a repo rule more often is probably not a breaking change. But it might be surprising?
        - Notably, we do a similar thing for `FileValue`, if we ever try to turn a `Label` into a `Path`. That is, if we (for example) try to read the

contents of a file identified by a label, we "watch" the file, and rerun the repo rule if the file changes. But we do *not* try to "watch" any file that's not pointed to by a label (so if we just tried to `rctx.read(rctx.path("/etc/myconfig.rc"))`, then changes to that file does not cause us to refetch). Why is that? Should we change that behavior to always watch those files?