Purpose

The mach command that is the basis for 93 different Firefox development commands requires Python 2. Python 2 has reached the official end of it's life. We need to move the mach toolchain to Python 3.

Facts and Considerations

- Mercurial is using Python 2. Python 3 support is in progress.
- Python 2 EOL is 2020.
- The Firefox build process is already Python 3.5.
- six.py is already in the Firefox source tree for use by mach.
- Downstream builders and packagers of Firefox, such as Linux distros, may not switch their build systems from Python 2 to Python 3 right away. It's a community-driven effort. e.g. Arch Linux (fast adopter) vs RHEL and Ubuntu LTS (long-life stable releases) vs Debian (community driven).
- The mozbuild module may disappear in 2019H2/2020H1 if we switch to a third-party build system such as Starlark.
- "mach bootstrap" and bootstrap.py needs to support Python 2 for a while (at least 2020) if we want a push-button contributor setup experience. e.g. macOS ships with Python 2 built-in, not Python 3.
- The Firefox CI test suite does not exercise mach or the developer setup experience.
- There are 93 mach commands. It will take a while to cull them and port them.
- Not all commands are used by developers. We have telemetry to show this.
- Porting is likely to increase Python Unicode bug reports.

Goals

- Gradual switch from 2 to 3.
- Push-button contributor bootstrap:
    - Support Python 2 bootstrap until at least 2020, maybe 2021 (depends on the macOS laptop lifecycle/developer population).
    - Support Python 3 bootstrap for when Python 2 disappears (this is already happening in some Linux distros).
- Increase safety of porting: set up automated testing of bootstrap, possibly other happy-path commands (maybe to "mach configure"?).
- Don't spend effort porting commands that aren't in use.
- Make mach development friendly to outside contributors and intern programs.

People

- kmoir: build team manager
- mshal, chmanchester: current maintainers. mach-core, mach-build, mach-bootstrap domain experts

- glob: Lvl 5 tech lead, long-time mach+vct domain expert

Rough Backlog

Contributor Process and Documentation

The porting effort will take a lot of time with contribution from other teams, interns, possibly outside contributors. We need to make sure we have a smooth story for those people to get set up for development, finish a task, have the task reviewed, tested a lot so it doesn't break the engineering population, and deployed.

- We will be done when: a new contributor can get set up, port part of mach from 2 to 3, get a review, and depoly the changes will very little help from the core team.

Port bootstrap to run under both Python 2 and Python 3.

"python bootstrap.py" and "mach bootstrap" needs to run on both Python 2 and Python 3 systems for the foreseeable future.

- We need to make sure the main module runs under both Python 2 and Python 3. Porting is a well-defined process.
- We will want an automated integration test, possibly continuous integration, to test changes on multiple Python versions. tox can help with this.
- We need to pick a target Python 3 version - possibly 3.5 because that's what the build tools use.
- We will be done when: the dev environment setup steps in the Firefox Developer Guide should work the same with all supported Python versions.

Port libraries in firefox:///third_party/python to run on both Python 2 and Python 3

"python bootstrap.py" and "mach bootstrap" needs to run on both Python 2 and Python 3 systems for the foreseeable future. We need to make sure any dependencies it pulls in from the Firefox source code's /third_party/python directory run under both Python 2 and Python 3.

- Porting is a well-defined process.
- The dependencies need to be tackled on a case-by-case basis: either the library is py3-ready, upgraded, ported, removed, or we carry both python2 and python3 versions of each.
- We need to pick a target Python 3 version - possibly 3.5 because that's what the build tools use.
- We will be done when: the dev environment setup steps in the Firefox Developer Guide should work the same with all supported Python versions.

Continuous Integration for bootstrap code changes

At this time of this writing “mach bootstrap” has to be tested manually on each development OS we support. The porting effort would be much easier if automated cross-platform end-to-end tests were in place for the “mach bootstrap” command. Ideas include:

- Using TaskCluster to run “mach bootstrap” through to “mach configure” on clean OS images
- Enlist help from our developer community to make mach bootstrap acceptance tests trivial for firefox developers run on their target OS when we cut a mach release.
- We will be done when: any changes to mach-core, mach bootstrap, or bootstrap.py are automatically acceptance-tested without no effort from the developer on all supported OSes.

Delete unused mach commands

We don’t want to spend time porting mach commands that are not being used by developers.

- We can check the mach command telemetry using the build tools dashboard to see which commands are least used by the Firefox developer population.
- Whomever takes the role of mach Product Owner can help determine which commands to keep and cut, and if the commands should be removed from the codebase or moved into a contrib/ directory.
- Mach may also want to grow a command extension system, similar to Git’s command extension system, if commands are to be moved out of tree but kept available to developers to install and maintain on their own.

Port mach-core to run under both Python 2 and Python 3

mach commands rely on mach-core to run. mach needs to run on both Python 2 and Python 3 systems for the foreseeable future.

- Porting is a well-defined process.
- The dependencies need to be tackled on a case-by-case basis: either the library is py3-ready, upgraded, ported, removed, or we carry both python2 and python3 versions of each.
- We will want an automated integration test, possibly continuous integration, to test changes on multiple Python versions. tox can help with this.
- We need to pick a target Python 3 version - possibly 3.5 because that’s what the build tools use.
- We will be done when: ???

Port ancillary mozbuild scripts to Python 3

mozbuild needs to have its roadmap defined, but there are a number of ancillary build scripts it calls out to that are owned by various Mozilla teams that will also need porting. Those scripts need to be ported to run on Python 3.

- Needs Product Owner/long-term roadmap input.
- python/mozbuild/mozbuild/action has a bunch of scripts we control that can be ported right away.
- We will be done when: ???

Add an interpreter-switching mechanism to mach-core so subcommands can run either Python 2 or Python 3

If we add a registry of mach sub-commands we can annotate that registry with the desired interpreter that sub-command should run under. That way we can have both Python 2 and Python 3 commands live side-by-side. We can port the Python 2 commands piecemeal.

- ahal's design: the implementor should consult with him