This directory contains scripts for our continuous integration.

One important thing to keep in mind when reading the scripts here is that they are all based off of Docker images, which we build for each of the various system configurations we want to run on Jenkins. This means it is very easy to run these tests yourself:

1. Figure out what Docker image you want. The general template for our images look like: `registry.pytorch.org/pytorch/pytorch-$BUILD_ENVIRONMENT:$DOCKER_VERSION`, where `$BUILD_ENVIRONMENT` is one of the build environments enumerated in pytorch-dockerfiles. The dockerfile used by jenkins can be found under the `.circle` directory

2. Run `docker run -it -u jenkins $DOCKER_IMAGE`, clone PyTorch and run one of the scripts in this directory.

The Docker images are designed so that any "reasonable" build commands will work; if you look in build.sh you will see that it is a very simple script. This is intentional. Idiomatic build instructions should work inside all of our Docker images. You can tweak the commands however you need (e.g., in case you want to rebuild with DEBUG, or rerun the build with higher verbosity, etc.).

We have to do some work to make this so. Here is a summary of the mechanisms we use:

- We install binaries to directories like `/usr/local/bin` which are automatically part of your PATH.
- We add entries to the PATH using Docker ENV variables (so they apply when you enter Docker) and `/etc/environment` (so they continue to apply even if you sudo), instead of modifying `PATH` in our build scripts.
- We use `/etc/ld.so.conf.d` to register directories containing shared libraries, instead of modifying `LD_LIBRARY_PATH` in our build scripts.
- We reroute well known paths like `/usr/bin/gcc` to alternate implementations with `update-alternatives`, instead of setting `CC` and `CXX` in our implementations.