Run Syntax Matcher

1. After the FR_clang image is ready, get into the clang directory of program binutils-fuzz_disassemble, by
   `cd programs/binutils-fuzz_disassemble/clang`
2. Create a docker container, by
   `docker run -ti --name=fr_binutils-fuzz_disassemble_clang -v $PWD:/src fr_clang bash`
   The -v options mounts [path/to/FixReverter]/programs/binutils-fuzz_disassemble/clang directory on the host machine onto the /src directory in the container.
3. Now you are at the /src directory of the docker container with a terminal. Setup binutils-fuzz_disassemble with
   `bash setup.sh`
4. Then build binutils-fuzz_disassemble with bear to generate the compilation database
   `bash build.sh`
   Linking errors are expected here.
5. Run the syntax matcher with
   `python3 /fixreverter/FixReverter/drivers/inject/driver.py -p`
   There will be an `apm.json` file storing the syntax match results im /src/tmp(mounted on programs/binutils-fuzz_disassemble/clang/tmp).
6. Detach from the clang tools container with CTRL+p + CTRL+q

Run Semantic Matcher

1. Go to the semantic matcher folder by
   `cd ../phasar`
   And copy apm.json file, which is the input for the semantic matcher
   `cp ../clang/tmp/apm.json .`
2. Start the docker container for the semantic matcher. Similarly, [path/to/FixReverter]/programs/binutils-fuzz_disassemble/phasar directory on the host machine is mounted on the /src directory in the container.
   `docker run -ti --name=fr_binutils-fuzz_disassemble_phasar -v $PWD:/src fr_phasar_port bash`
3. Download and build program for the semantic matcher
   `bash build.sh`
4. Run the semantic matcher
   `bash run_phasar.sh`
   For some programs this step may take up to 200GB memory. The process will end with an error when it runs out of memory. It can take up to 3 days.
   After the process finishes, the output file is stored at /src/out/dda.json.
5. Now we can stop the fr_binutils-fuzz_disassemble_phasar container with CTRL+D

Run Injector and Naive Bug Filter

1. Go back to the clang folder
   `cd ../clang`
   And copy the semantic matcher output from the previous step, or the provided one
   `cp ../phasar/out/dda.json .`
   Or
   `cp ../inject_products/dda.json .`
   Then go back to the fr_binutils-fuzz_disassemble_clang container
   `docker attach fr_binutils-fuzz_disassemble_clang`
2. Move the semantic matcher output to the correct location.
   `mv ./dda.json ./tmp`
   This step is done in the docker container to avoid permission issues.
3. Rewrite the program
   `python3 /fixreverter/FixReverter/drivers/inject/driver.py -i`
4. Turn off LeakSanitizer
   `export ASAN_OPTIONS=detect_leaks=0`
5. Clean the previous build
   `cd binutils-gdb && make distclean && find . -type f -name "*config.cache" -delete && cd ..`
6. Build the coverage binary
   `bash build_cov.sh`
   Errors are expected on this step.
7. Run NaiveBugFilter and re-inject with filtered bugs
   `python3 /fixreverter/FixReverter/drivers/inject/driver.py -r`
8. Rebuild the program in order to allow some commands in the build script to take effect
   `bash build_cov.sh`
9. Get the diff for the injection by
   `cd binutils-gdb`
   `make distclean && find . -type f -name "*config.cache" -delete`
   `git diff >> fr_injection.patch`
   Now the binutils-fuzz_disassemble program is ready for fuzzing. See the next section, *Run FuzzBench Experiment* on how to fuzz it.
10. All 3 intermediate products, the apm.json, dda.json and inject.json are provided in programs/binutils-fuzz_disassemble/inject_products.