

## Run Syntax Matcher

1. After the FR\_clang image is ready, get into the clang directory of program proj4, by  
`cd programs/proj4/clang`
2. Create a docker container, by  
`docker run -ti --name=fr_proj4_clang -v $PWD:/src fr_clang bash`  
The -v options mounts [path/to/FixReverter]/programs/proj4/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 proj4 with  
`bash setup.sh`
4. Then build proj4 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 in /src/tmp(mounted on programs/proj4/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/proj4/phasar directory on the host machine is mounted on the /src directory in the container.  
`docker run -ti --name=fr_proj4_phasar -v $PWD:/src fr_phasar 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\_proj4\_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\_proj4\_clang container

```
docker attach fr_proj4_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. Clean the previous build

```
cd PROJ && make distclean && cd ..
```

4. Rewrite the program

```
python3 /fixreverter/FixReverter/drivers/inject/driver.py -i
```

5. Turn off LeakSanitizer

```
export ASAN_OPTIONS=detect_leaks=0
```

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. Get the diff for the injection by

```
cd PROJ
```

```
make distclean
```

```
git diff >> fr_injection.patch
```

Now the proj4 program is ready for fuzzing. See the next section, *Run FuzzBench Experiment* on how to fuzz it.

9. All 3 intermediate products, the apm.json, dda.json and inject.json are provided in programs/proj4/inject\_products.