NobleProg

Lab Exercise 4 – C++ Build using Bazel

(Multiple Packages, Multiple Targets)

This next stage adds another layer of complication and builds a project with multiple packages. Take a look below at the structure and contents of the NobleProg3 directory:

```
└──NobleProg3
   ├── main
   │   ├── BUILD
   │   ├── hello-world.cc
   │   ├── hello-greet.cc
   │   └── hello-greet.h
   ├── lib
   │   ├── BUILD
   │   ├── hello-time.cc
   │   └── hello-time.h
   └── WORKSPACE
```

You can see that now there are two sub-directories, and each contains a BUILD file. Therefore, to Bazel, the workspace now contains two packages: lib and main.

Take a look at the lib/BUILD file:

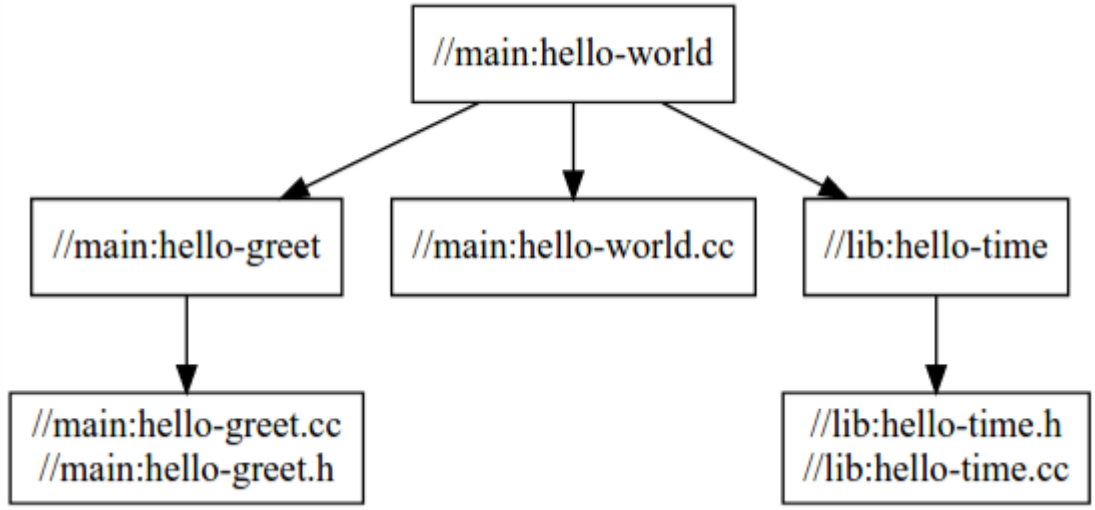
```
cc_library(
    name = "hello-time",
    srcs = ["hello-time.cc"],
    hdrs = ["hello-time.h"],
    visibility = ["//main:__pkg__"],
)
```

And at the main/BUILD file:

```
cc_library(
    name = "hello-greet",
    srcs = ["hello-greet.cc"],
    hdrs = ["hello-greet.h"],
)

cc_binary(
    name = "hello-world",
    srcs = ["hello-world.cc"],
    deps = [
        ":hello-greet",
        "//lib:hello-time",
    ],
)
```

The hello-world target in the main package depends on the hello-time target in the lib package (hence the target label //lib:hello-time) - Bazel knows this through the deps attribute. You can see this reflected in the dependency graph:



For the build to succeed, you make the //lib:hello-time target in lib/BUILD explicitly visible to targets in main/BUILD using the visibility attribute. This is because by default targets are only visible to other targets in the same BUILD file. Bazel uses target visibility to prevent issues such as libraries containing implementation details leaking into public APIs.

Now build this final version of the project. Switch to the cpp-tutorial/stage3 directory by running:

```
cd  ../NobleProg3
```

Once again, run the following command:

```
bazel build //main: hello-world
```

Bazel produces something that looks like this:

```
INFO: Found 1 target...
Target //main:hello-world up-to-date:
  bazel-bin/main/hello-world
INFO: Elapsed time: 0.167s, Critical Path: 0.00s
```

Now test the last binary of this tutorial for a final Hello world message:

```
bazel-bin/main/hello-world
```

Summary: NobleProg 3

You've now built the project as two packages with three targets and understand the dependencies between them, which equips you to go forth and build future projects with Bazel. In the next section, take a look at how to continue your Bazel journey.