Lab Exercise 6.2 – C++ Build using Bazel

(Sharing Variables using args and Env_Var)

Sharing variables in Bazel builds using C++ can be accomplished by passing values as command-line arguments or by using environment variables. Here are two common methods for sharing variables in a Bazel build:

Method 1: Using Command-Line Arguments

Define a Variable in a Build Rule: In your BUILD file, you can define a variable as part of a build rule and then pass it as a command-line argument to the binary target. Here's an example:

```
cc_binary(
  name = "my_program",
  srcs = ["main.cpp"],
  args = [
    "--my_variable=value_to_share",
  ],
)
```

Access the Variable in C++ Code: In your C++ code (e.g., main.cpp), you can access the command-line arguments to retrieve the shared variable:

```
#include <iostream>
int main(int argc, char** argv) {
  for (int i = 0; i < argc; ++i) {
    std::cout << "Argument " << i << ": " << argv[i] << std::endl;
  }

  // Access the shared variable by its position in argv
  if (argc > 1) {
    std::string sharedVariable = argv[1];
```

```
        std::cout << "Shared Variable: " << sharedVariable << std::endl;
    }

    return 0;
}
```

Build and Run the Program: Build your program using Bazel and then run it. The shared variable is passed as a command-line argument.

Method 2: Using Environment Variables

Define an Environment Variable in a BUILD File: In your BUILD file, you can define an environment variable using the env attribute of a build rule:

```
cc_binary(
    name = "my_program",
    srcs = ["main.cpp"],
    env = {
        "MY_VARIABLE": "value_to_share",
    },
)
```

Access the Environment Variable in C++ Code: In your C++ code (e.g., main.cpp), you can access the environment variable using the getenv function:

```
#include <iostream>
#include <cstdlib> // For getenv
int main() {
    const char* sharedVariable = std::getenv("MY_VARIABLE");
    if (sharedVariable) {
        std::cout << "Shared Variable: " << sharedVariable << std::endl;
    } else {
        std::cout << "Shared Variable not found." << std::endl;
    }
    return 0;
}
```

Build and Run the Program: Build your program using Bazel and then run it. The shared variable is accessed from the environment.

Choose the method that best suits your project's needs. Using command-line arguments is straightforward and allows for flexibility in passing values, while environment variables are more suitable for configuration settings that need to be available throughout the program's execution.