CGo Recursive Copy

Functionality

When you execute the command `copy_recursive_bin *<source directory>* *<destination directory>*`, the process will conduct a recursive copy of the source directory to the destination. Each step of the copy operation will be asynchronously recorded in a log file.

If, for any reason, the logger fails to function, the process will proceed with the copy but will notify the user via their terminal that logging is not operational.

Make it, Launch it

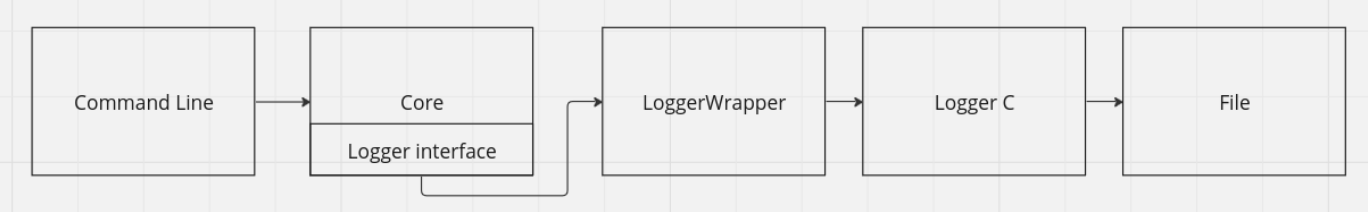
In order to test this program you need to have install you GO environment and then just do:
1. `Make`

2. `./bin/copy_recursive_bin *<source directory>* *<destination directory>*`

Architecture

I've chosen a hexagonal architecture for this project, where the core is responsible for handling the functionality's logic. Within this core, there's an interface called Logger that facilitates the injection of logger dependencies.

This architectural decision was made to provide flexibility in changing the logger's implementation without necessitating modifications to the core code of the project.



Improvement

1. **Decentralized Logging: Leveraging Message Publishing for Enhanced System Flexibility**

   In this context, the logger functions as a C-written code, invoked by the core. Should there be a desire to externalize the logging system, an alternative approach could involve publishing a message through communication channels such as SNS, Kafka, or RabbitMQ. Subsequently, a separate service would be tasked with listening to the designated topic and generating logs based on the received message.

2. **Optimizing Command Line Functionality**

   If we aim to enhance command line parameters, such as introducing the option to copy a file instead of a directory, we can leverage the `flag` package. This package facilitates the definition and management of flags in a command line interface.

3. **Implementation of tests**

   Here, we could implement unit tests to verify that every function is called at the right time and to ensure that Go routines are used appropriately. Additionally, the implementation of end-to-end tests would be beneficial to verify if the functionality's goal has been achieved, such as confirming the creation of the destination directory with the correct permissions.

4. **Security**

   In this coding exercise, we employ a basic command line for directory copying. Consider a scenario where, in the future, we substitute this command line with HTTP handlers. This transition would enable us to introduce context to the request, allowing us to scrutinize the request's author and ascertain whether they possess the necessary permissions, possibly through mechanisms like OAuth.