Concurrent Programming: Languages and Techniques

Channel-based Concurrency Module
Lab 4: Project Support

13 October 2022

MIEI - Integrated Masters in Comp. Science and Informatics
Specialization Block

Bernardo Toninho
(with António Ravara and Carla Ferreira)

NOVALINCS

Building Things Concurrently

- Some clarifications:
  - Graph nodes correspond to goroutines (i.e. one goroutine responsible for each file).
  - The “coordinator”, sends a “dummy” message to the **leaves** of the graph/tree to signal the start of a build round.
  - The “coordinator” is notified the round ended when it receives a message from the **root** of the tree/graph.

Building Things Concurrently

- Some clarifications:
  - The graph can/should be built sequentially…
  - The goal is to setup the nodes and their connections correctly.
  - Each node must be able to **receive** notifications from its dependencies.
  - Each node must be able to **send** notifications to those that depend on it.

Building Things Concurrently

- Traversing the dependency file and making the graph:
  - Map data structure maps **objects** (dep file contents) to “nodes”.
  - A node has the target and its dependency names, whether its already “defined” or not, being visited, and maybe some more useful data… (e.g. channels to talk with its successors)
  - Initially, populate the map with “undefined” states (i.e., the file contents).
  - A node is only defined when its corresponding thread and all its antecedents are created and linked.

Building Things Concurrently

- Traversing the dependency file and making the graph:
  - An **internal node** is “undefined” when it has not yet been encountered during the traversal.
  - A **leaf** is “undefined” when not yet encountered. This means it won’t be in the map at all!
  - After initializing the map, apply a `process` function to the **target** object of each rule.

Building Things Concurrently

- `Process`-ing a `target`:
  - Lookup the `target` in the map:
    - If absent from the map, its a leaf — make it (spawn its thread, etc.) and update the map accordingly.
    - If undefined, `visit` it.
    - If defined, nothing left to do.

Building Things Concurrently

- `Visit`-ing a `target`:
  - `Process` the target's dependencies.
  - Now you can create the goroutine responsible for this target, its channel(s), because the map will have been populated with this data for its dependencies.
  - That's (mostly) it…

Building Things Concurrently

- Things to think about:
  - A node must send to (potentially) many other nodes (that depend on it) — how to achieve that?
  - A node must receive from many other nodes (its dependencies) — how to achieve that?
  - If the map stores “the” output channel for each node, it solves the second half of the problem, but not the first.
  - Now its up to you… :)