|  |
| --- |
| Circle Language Spec: Commands |

## Recursion

Recursion in commands happens when a command contains a call to the same command definition again.

It can also occur when one command calls another command and the other command potentially calls the first command again.

In fact, in any string of commands, that somewhere along the line calls the first command again, all the commands in the string are called recursive.

The *point* of recursive execution is that it is not endlessly recursive after all. At some point, a potential execution should not be executed, in order to exit the seemingly endless recursion.

The problem with endless recursion is solved, by the fact, that private contents of an executable object are only created just before it is actually going to execute.

When a command definition contains a call to the same command definition, then the call does not contain any private content yet, because it is not executing yet.

When a command call contains a call to the same command definition, then the call also does not contain any private content yet, before it executes.

Only when the call actually executes, you can see private contents inside the recursive call. So only when recursive calls are actually busy executing, this recursive repetition is present in the system.

So only creating private content just before a command actually executes solves the problem of endless recursion.

# Ideas

# Out of the original Symbol documentation

* Recursive calls... hmmm... advanced issue. Hoef je je klomp niet over te breken.
  + Recursive calls... hmmm... advanced issue. Hoef je je klomp niet over te breken.

< Maar lijkt een Execution basics. Hmmm… de term Execution Basics is niet goed genoeg dan. De term basics is niet goed. Execution Facts… nah.