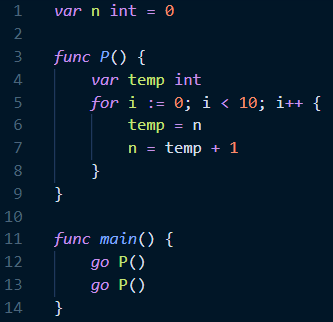
# Concurrent counting example

Demonstrated by Ýmir Þórleifsson & Bjartur Þórhallsson

Code snippet shows that the *P* function is run by two threads concurrently, each reading and writing a shared integer variable, *n*. *P* increments the value of *n* by ten.

We assume there is no way to know in what order each thread is executed so we have an interleaving of possibilities.

The most important lines in the code snippet are lines 6 and 7, there the threads are read and writing the shared variable n. Our goal is for n to be equal to 2 when both threads terminate.

P0 is the first thread, P1 is the second thread.

We start by executing both threads, they both read 0 for n from memory. Then we execute P0 until it finishes its second last iteration, now we switch over to P1, which has n = 0 saved, and run it through one iteration which makes n = 1 before we switch to P0 again to save the new n = 1. Now we let P1 run until it finishes execution, and we switch over to P0, increment n = 1 by one and finish execution, resulting in n = 2 at termination.

See the table on the next page for the interleaving.

|  |  |  |  |
| --- | --- | --- | --- |
| P0 | P1 | *n* | *Comments* |
| 4 |  | 0 |  |
|  | 4 | 0 |  |
| 5 |  | 0 |  |
|  | 5 | 0 |  |
| 6 |  | 0 | P0 gets n from memory. |
|  | 6 | 0 | P1 gets n from memory. |
| 7 |  | 1 | P0 executes until it finishes the second last iteration. |
| 6 |  | 2 |  |
| 7 |  | 3 |  |
| ... |  | ... |  |
| 6 |  | 8 |  |
| 7 |  | 9 |  |
|  | 7 | 1 | When P1 resumes its execution, it increments n which it read as 0 by one and writes it to memory. |
| 6 |  | 1 | P0 reads n from memory which is no 1. |
|  | 6 | 1 | P1 now runs until it finishes execution. |
|  | 7 | 2 |  |
|  | 6 | 2 |  |
|  | ... | ... |  |
|  | 6 | 8 |  |
|  | 7 | 9 |  |
| 7 |  | 2 | P0 resumes for its last line and increments the n it has saved, which is 1 by 1 and saves it to memory. |
|  | fin |  | P1 exits the for-loop and P. |
| fin |  |  | P0 exits the for-loop and P. |