

```

┌────────────────────────── MODULE init_once ───────────────────────────┐
EXTENDS TLC, FiniteSets, Integers

CONSTANTS Processes

--algorithm init_once
variables
  lock = FALSE;
  is_init = FALSE;

  pids = {};

define
  just_once  $\triangleq \Diamond(\text{Cardinality}(\textit{pids}) = 1) \wedge \Box(\text{Cardinality}(\textit{pids}) \leq 1)$ 
end define

initializer
fair + process pid  $\in$  Processes
begin
  start_init_once:
    while  $\neg \textit{is\_init}$  do
      load_lock_relaxed:
        if  $\neg \textit{lock}$  then
          compare_exchange:
            if  $\neg \textit{lock}$  then
              lock := TRUE;
              pids := pids  $\cup$  {self};

              initialize:
                skip;

              store_is_init:
                is_init := TRUE;
            end if ;
          end if ;
        end while ;
      end process ;
    end algorithm ;

  BEGIN TRANSLATION (chksum(pcal) = "f8df04e6"  $\wedge$  chksum(tla) = "153edf40")
  VARIABLES lock, is_init, pids, pc

  define statement
    just_once  $\triangleq \Diamond(\text{Cardinality}(\textit{pids}) = 1) \wedge \Box(\text{Cardinality}(\textit{pids}) \leq 1)$ 

    vars  $\triangleq \langle \textit{lock}, \textit{is\_init}, \textit{pids}, \textit{pc} \rangle$ 

    ProcSet  $\triangleq$  (Processes)

```

$$\begin{aligned}
Init &\triangleq \text{Global variables} \\
&\wedge lock = \text{FALSE} \\
&\wedge is_init = \text{FALSE} \\
&\wedge pids = \{\} \\
&\wedge pc = [self \in ProcSet \mapsto \text{"start_init_once"}] \\
\\
start_init_once(self) &\triangleq \wedge pc[self] = \text{"start_init_once"} \\
&\wedge \text{IF } \neg is_init \\
&\quad \text{THEN } \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"load_lock_relaxed"}] \\
&\quad \text{ELSE } \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"Done"}] \\
&\wedge \text{UNCHANGED } \langle lock, is_init, pids \rangle \\
\\
load_lock_relaxed(self) &\triangleq \wedge pc[self] = \text{"load_lock_relaxed"} \\
&\wedge \text{IF } \neg lock \\
&\quad \text{THEN } \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"compare_exchange"}] \\
&\quad \text{ELSE } \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"start_init_once"}] \\
&\wedge \text{UNCHANGED } \langle lock, is_init, pids \rangle \\
\\
compare_exchange(self) &\triangleq \wedge pc[self] = \text{"compare_exchange"} \\
&\wedge \text{IF } \neg lock \\
&\quad \text{THEN } \wedge lock' = \text{TRUE} \\
&\quad \wedge pids' = (pids \cup \{self\}) \\
&\quad \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"initialize"}] \\
&\quad \text{ELSE } \wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"start_init_once"}] \\
&\quad \wedge \text{UNCHANGED } \langle lock, pids \rangle \\
&\wedge \text{UNCHANGED } is_init \\
\\
initialize(self) &\triangleq \wedge pc[self] = \text{"initialize"} \\
&\wedge \text{TRUE} \\
&\wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"store_is_init"}] \\
&\wedge \text{UNCHANGED } \langle lock, is_init, pids \rangle \\
\\
store_is_init(self) &\triangleq \wedge pc[self] = \text{"store_is_init"} \\
&\wedge is_init' = \text{TRUE} \\
&\wedge pc' = [pc \text{ EXCEPT } ![self] = \text{"start_init_once"}] \\
&\wedge \text{UNCHANGED } \langle lock, pids \rangle \\
\\
pid(self) &\triangleq start_init_once(self) \vee load_lock_relaxed(self) \\
&\vee compare_exchange(self) \vee initialize(self) \\
&\vee store_is_init(self) \\
\\
\text{Allow infinite stuttering to prevent deadlock on termination.} \\
Terminating &\triangleq \wedge \forall self \in ProcSet : pc[self] = \text{"Done"} \\
&\wedge \text{UNCHANGED } vars \\
\\
Next &\triangleq (\exists self \in Processes : pid(self)) \\
&\vee Terminating
\end{aligned}$$

$$\begin{aligned}
Spec &\triangleq \wedge Init \wedge \Box [Next]_{vars} \\
&\quad \wedge \forall self \in Processes : SF_{vars}(pid(self)) \\
Termination &\triangleq \Diamond (\forall self \in ProcSet : pc[self] = \text{"Done"})
\end{aligned}$$

END TRANSLATION
