

---

MODULE *init\_once*

---

EXTENDS *TLC, FiniteSets, Integers*

CONSTANTS *Processes*

**--algorithm** *init\_once*

**variables**

*lock* = FALSE;

*is\_init* = FALSE;

*pids* = {};

**define**

*just\_once*  $\triangleq$  *Cardinality(pids) ≤ 1*

**end define**

    initializer

**fair + process** *pid* ∈ *Processes*

**begin**

*start\_init\_once*:

**while**  $\neg is\_init$  **do**

*load\_lock\_relaxed*:

**if**  $\neg lock$  **then**

*compare\_exchange*:

**if**  $\neg lock$  **then**

*lock* := TRUE;

*pids* := *pids* ∪ {*self*};

*initialize*:

**skip**;

*store\_is\_init*:

*is\_init* := TRUE;

**end if** ;

**end if** ;

**end while** ;

**end process** ;

**end algorithm** ;

    BEGIN TRANSLATION (*chksum(pcal)* = "ded479d9" ∧ *chksum(tla)* = "ae662f87")

VARIABLES *lock, is\_init, pids, pc*

    define statement

*just\_once*  $\triangleq$  *Cardinality(pids) ≤ 1*

*vars*  $\triangleq$  ⟨*lock, is\_init, pids, pc*⟩

*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

---