MODULE *Simple*

See the paper "Teaching Concurrency" by *Leslie Lamport* for the problem (https://*www.microsoft.com*/en-us/research/uploads/prod/2016/12/Teaching-*Concurrency.pdf*).

See also the *StackOverflow* post "What is the inductive invariant of the simple concurrent program?" (https://*stackoverflow.com*/*q*/24989756/1833118).

See the answer (https://*stackoverflow.com*/a/46108331/1833118) to the post above for the TLA+ specification and *TLAPS* proof.

EXTENDS *Integers*, *TLAPS*

CONSTANTS N the number of processes

```
--algorithm Simple

variables
    x = [i ∈ 0 .. N − 1 ↦ 0] ;
    y = [i ∈ 0 .. N − 1 ↦ 0] ;

process Proc ∈ 0 .. N − 1
begin
    s1: x[self] := 1 ;
    s2: y[self] := x[(self − 1)%N]
end process

end algorithm
```

BEGIN TRANSLATION

VARIABLES x, y, pc

$vars \triangleq \langle x, y, pc \rangle$

$ProcSet \triangleq (0 \,..\, N-1)$

$Init \triangleq$ Global variables
$\land\ x = [i \in 0 \,..\, N-1 \mapsto 0]$
$\land\ y = [i \in 0 \,..\, N-1 \mapsto 0]$
$\land\ pc = [self \in ProcSet \mapsto \text{"s1"}]$

$s1(self) \triangleq \land\ pc[self] = \text{"s1"}$
$\land\ x' = [x \text{ EXCEPT } ![self] = 1]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"s2"}]$
$\land\ y' = y$

$s2(self) \triangleq \land\ pc[self] = \text{"s2"}$
$\land\ y' = [y \text{ EXCEPT } ![self] = x[(self - 1)\%N]]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"Done"}]$
$\land\ x' = x$

```
54  Proc(self) ≜ s1(self) ∨ s2(self)

56   Allow infinite stuttering to prevent deadlock on termination.
57  Terminating ≜ ∧ ∀ self ∈ ProcSet : pc[self] = "Done"
58                ∧ UNCHANGED vars

60  Next ≜ (∃ self ∈ 0 .. N − 1 : Proc(self))
61           ∨ Terminating

63  Spec ≜ Init ∧ □[Next]_vars

65  Termination ≜ ◇(∀ self ∈ ProcSet : pc[self] = "Done")

67   END TRANSLATION
68 ──────────────────────────────────────────────────────────────────────
69  AtLeastOneYWhenDone ≜ (∀ i ∈ 0 .. N − 1 : pc[i] = "Done") ⇒ ∃ i ∈ 0 .. N − 1 : y[i] = 1

71  TypeOK ≜
72      ∧   x ∈ [0 .. N − 1 → {0, 1}]
73      ∧   y ∈ [0 .. N − 1 → {0, 1}]
74      ∧   pc ∈ [ProcSet  → {"s1", "s2", "Done"}]

76  Inv ≜
77      ∧ TypeOK
78      ∧ ∀ i ∈ 0 .. N − 1 : (pc[i] ∈ {"s2", "Done"} ⇒ x[i] = 1)
79      ∧ AtLeastOneYWhenDone
80 ──────────────────────────────────────────────────────────────────────
81  ASSUME NIsInNat ≜ N ∈ Nat \ {0}

83   TLAPS doesn't know this property of modulus operator
84  AXIOM ModInRange ≜ ∀ i ∈ 0 .. N − 1 : (i − 1)%N ∈ 0 .. N − 1

86  THEOREM Spec ⇒ □AtLeastOneYWhenDone
87  ⟨1⟩ USE DEF ProcSet, Inv
88  ⟨1⟩1. Init ⇒ Inv
89      BY NIsInNat DEF Init, Inv, TypeOK, AtLeastOneYWhenDone
90  ⟨1⟩2. Inv ∧ [Next]_vars ⇒ Inv′
91    ⟨2⟩ SUFFICES ASSUME Inv, [Next]_vars
92                 PROVE  Inv′
93      OBVIOUS
94    ⟨2⟩1.CASE Next
95      ⟨3⟩1.CASE ∃ self ∈ 0 .. N − 1 : Proc(self)
96        ⟨4⟩ SUFFICES ASSUME NEW self ∈ 0 .. N − 1, Proc(self)
97                     PROVE  Inv′
98          BY ⟨3⟩1
99        ⟨4⟩1.CASE s1(self)
100         BY ⟨4⟩1, NIsInNat DEF s1, TypeOK, AtLeastOneYWhenDone
101       ⟨4⟩2.CASE s2(self)
```

```
102          BY ⟨4⟩2, NIsInNat, ModInRange DEF s2, TypeOK, AtLeastOneYWhenDone
103        ⟨4⟩3. QED
104          BY ⟨3⟩1, ⟨4⟩1, ⟨4⟩2  DEF Proc
105      ⟨3⟩2.CASE (∀ self ∈ ProcSet : pc[self] = "Done") ∧ UNCHANGED vars
106          BY ⟨3⟩2  DEF TypeOK, vars, AtLeastOneYWhenDone
107      ⟨3⟩3. QED
108        BY ⟨2⟩1, ⟨3⟩1, ⟨3⟩2  DEF Next
109    ⟨2⟩2.CASE UNCHANGED vars
110      BY ⟨2⟩2  DEF TypeOK, vars, AtLeastOneYWhenDone
111    ⟨2⟩3. QED
112      BY ⟨2⟩1, ⟨2⟩2
113  ⟨1⟩3. Inv ⇒ AtLeastOneYWhenDone
114      OBVIOUS
115  ⟨1⟩4. QED
116      BY ⟨1⟩1, ⟨1⟩2, ⟨1⟩3, PTL DEF Spec
117 ================================================================================
```

\ ∗ Modification History
\ ∗ Last modified *Fri Aug* 02 14:30:22 *CST* 2019 by *hengxin*
\ ∗ Created *Fri Aug* 02 13:28:48 *CST* 2019 by *hengxin*