MODULE *scheduler*

EXTENDS *TLC*, *Sequences*, *SequencesExt*, *FiniteSets*

CONSTANTS *Subscribers*, *Timers*, *Workers*

$AllTask \triangleq Subscribers \cup Subscribers$

```
--algorithm scheduler
variables
    events
    wait_set = {} ;

    states of tasks
    run_queue = ⟨⟩ ;
    running = {} ;
    waiting  = AllTask ;

define
    starvation_free ≜ ∀ event ∈ AllTask : event ∈ wait_set ⇝ ◇(event ∈ running)
end define

fair  + process scheduler = "scheduler"
begin
    start_sched:
        while TRUE do
                await wait_set ≠ {} ;

                  pick runnable tasks and change the states to run_queue from waiting
                with tasks = waiting ∩ wait_set,
                     timers = tasks ∩ Timers,
                     subscribers = tasks ∩ Subscribers do
                        push to run_queue
                       run_queue := run_queue ∘ SetToSeq(timers) ∘ SetToSeq(subscribers) ;

                        change state
                       waiting := (waiting \ timers) \ subscribers ;
                end with  ;
        end while  ;
end process  ;

fair process trigger_subscriber ∈ Subscribers
begin
    start_subscriber:
        while TRUE do
            wait_set := wait_set ∪ {self} ;
        end while  ;
end process  ;
```

```
fair process trigger_timer ∈ Timers
begin
    start_timer:
        while TRUE do
            wait_set := wait_set ∪ {self} ;
        end while ;
end process ;

 worker thread
fair + process worker ∈ Workers
variables
    task ;
begin
     work-stealing
    start_worker:
        while TRUE do
            await run_queue ≠ ⟨⟩     ;

            task := Head(run_queue) ;
            run_queue := Tail(run_queue) ;
            running := running ∪ {task} ;

            finish_task:
                running := running \ {task} ;
                waiting  := waiting ∪ {task} ;
        end while ;
end process ;

end algorithm  ;
 BEGIN TRANSLATION (chksum(pcal) = "a5cfbdca" ∧ chksum(tla) = "534b1cd4")
CONSTANT defaultInitValue
VARIABLES wait_set, run_queue, running, waiting, pc

 define statement
starvation_free ≜ ∀ event ∈ AllTask : event ∈ wait_set ⇝ ◇(event ∈ running)

VARIABLE task

vars ≜ ⟨wait_set, run_queue, running, waiting, pc, task⟩

ProcSet ≜ {"scheduler"} ∪ (Subscribers) ∪ (Timers) ∪ (Workers)

Init ≜  Global variables
        ∧ wait_set = {}
        ∧ run_queue = ⟨⟩
        ∧ running = {}
        ∧ waiting  = AllTask
         Process worker
```

$$
\begin{array}{l}
\land\ task = [self \in Workers \mapsto defaultInitValue] \\
\land\ pc = [self \in ProcSet \mapsto \text{CASE } self = \text{“scheduler”} \to \text{“start\_sched”} \\
\qquad\qquad\qquad\qquad\qquad\quad \square\ \ self \in Subscribers \to \text{“start\_subscriber”} \\
\qquad\qquad\qquad\qquad\qquad\quad \square\ \ self \in Timers \to \text{“start\_timer”} \\
\qquad\qquad\qquad\qquad\qquad\quad \square\ \ self \in Workers \to \text{“start\_worker”}]
\end{array}
$$

$$
\begin{array}{l}
start\_sched \triangleq\ \land\ pc[\text{“scheduler”}] = \text{“start\_sched”} \\
\qquad\qquad\qquad\ \land\ wait\_set \neq \{\} \\
\qquad\qquad\qquad\ \land \text{LET } tasks \triangleq waiting \cap wait\_set \text{IN} \\
\qquad\qquad\qquad\qquad \text{LET } timers \triangleq tasks \cap Timers \text{IN} \\
\qquad\qquad\qquad\qquad\quad \text{LET } subscribers \triangleq tasks \cap Subscribers \text{IN} \\
\qquad\qquad\qquad\qquad\qquad \land\ run\_queue' = run\_queue \circ SetToSeq(timers) \circ SetToSeq(subscribers) \\
\qquad\qquad\qquad\qquad\qquad \land\ waiting' = (waiting \setminus timers) \setminus subscribers \\
\qquad\qquad\qquad\ \land\ pc' = [pc \text{ EXCEPT } ![\text{“scheduler”}] = \text{“start\_sched”}] \\
\qquad\qquad\qquad\ \land \text{UNCHANGED } \langle wait\_set,\ running,\ task \rangle
\end{array}
$$

$$
scheduler \triangleq start\_sched
$$

$$
\begin{array}{l}
start\_subscriber(self) \triangleq\ \land\ pc[self] = \text{“start\_subscriber”} \\
\qquad\qquad\qquad\qquad\qquad\ \land\ wait\_set' = (wait\_set \cup \{self\}) \\
\qquad\qquad\qquad\qquad\qquad\ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“start\_subscriber”}] \\
\qquad\qquad\qquad\qquad\qquad\ \land \text{UNCHANGED } \langle run\_queue,\ running,\ waiting,\ task \rangle
\end{array}
$$

$$
trigger\_subscriber(self) \triangleq start\_subscriber(self)
$$

$$
\begin{array}{l}
start\_timer(self) \triangleq\ \land\ pc[self] = \text{“start\_timer”} \\
\qquad\qquad\qquad\qquad\ \land\ wait\_set' = (wait\_set \cup \{self\}) \\
\qquad\qquad\qquad\qquad\ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“start\_timer”}] \\
\qquad\qquad\qquad\qquad\ \land \text{UNCHANGED } \langle run\_queue,\ running,\ waiting,\ task \rangle
\end{array}
$$

$$
trigger\_timer(self) \triangleq start\_timer(self)
$$

$$
\begin{array}{l}
start\_worker(self) \triangleq\ \land\ pc[self] = \text{“start\_worker”} \\
\qquad\qquad\qquad\qquad\ \land\ run\_queue \neq \langle\rangle \\
\qquad\qquad\qquad\qquad\ \land\ task' = [task \text{ EXCEPT } ![self] = Head(run\_queue)] \\
\qquad\qquad\qquad\qquad\ \land\ run\_queue' = Tail(run\_queue) \\
\qquad\qquad\qquad\qquad\ \land\ running' = (running \cup \{task'[self]\}) \\
\qquad\qquad\qquad\qquad\ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“finish\_task”}] \\
\qquad\qquad\qquad\qquad\ \land \text{UNCHANGED } \langle wait\_set,\ waiting \rangle
\end{array}
$$

$$
\begin{array}{l}
finish\_task(self) \triangleq\ \land\ pc[self]\ \ = \text{“finish\_task”} \\
\qquad\qquad\qquad\qquad\ \land\ running' = running \setminus \{task[self]\} \\
\qquad\qquad\qquad\qquad\ \land\ waiting'\ \ = (waiting \cup \{task[self]\}) \\
\qquad\qquad\qquad\qquad\ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“start\_worker”}] \\
\qquad\qquad\qquad\qquad\ \land \text{UNCHANGED } \langle wait\_set,\ run\_queue,\ task \rangle
\end{array}
$$

$$
worker(self) \triangleq start\_worker(self) \lor finish\_task(self)
$$

$$
\begin{array}{lll}
\mathit{Next} & \triangleq & \mathit{scheduler} \\
& & \quad \lor (\exists\, \mathit{self} \in \mathit{Subscribers} : \mathit{trigger\_subscriber}(\mathit{self})) \\
& & \quad \lor (\exists\, \mathit{self} \in \mathit{Timers} : \mathit{trigger\_timer}(\mathit{self})) \\
& & \quad \lor (\exists\, \mathit{self} \in \mathit{Workers} : \mathit{worker}(\mathit{self})) \\
\\
\mathit{Spec} & \triangleq & \land\, \mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}} \\
& & \land\, \mathrm{SF}_{\mathit{vars}}(\mathit{scheduler}) \\
& & \land\, \forall\, \mathit{self} \in \mathit{Subscribers} : \mathrm{WF}_{\mathit{vars}}(\mathit{trigger\_subscriber}(\mathit{self})) \\
& & \land\, \forall\, \mathit{self} \in \mathit{Timers} : \mathrm{WF}_{\mathit{vars}}(\mathit{trigger\_timer}(\mathit{self})) \\
& & \land\, \forall\, \mathit{self} \in \mathit{Workers} : \mathrm{SF}_{\mathit{vars}}(\mathit{worker}(\mathit{self}))
\end{array}
$$

END TRANSLATION