MODULE *heartbeat*

heartbeat application is a lightweight tool that executes and monitors a process. It restarts the process when it quits abnormally. It kills and starts it if it is unresponsive to a heartbeat message for a configured period of time. It sends the heartbeat messages over a *ZMQ* socket.

EXTENDS *Integers*, *Sequences*

CONSTANT
    *MaxEvents*    Maximum number of events.

VARIABLES
    *events*    Events that arrive from the checker threads.
  , *heartbeat*    The current heartbeat state of the process.
  , *process*    The current state of the process.
  , *signal*    The current *UNIX* signal state of the process.

$vars \triangleq \langle events, heartbeat, process, signal \rangle$

$AllEvents \triangleq \{\text{"timeout"}, \text{"aborted"}, \text{"complete"}, \text{"signaled"}\}$
$HeartbeatStates \triangleq \{\text{"ready"}, \text{"req"}, \text{"timeout"}\}$
$ProcessStates \triangleq \{\text{"ready"}, \text{"running"}, \text{"terminated"}, \text{"killed"}\}$
$SignalStates \triangleq \{\text{"ready"}, \text{"listening"}\}$

$TypeOK \triangleq$
$\quad \land \; events \in Seq(AllEvents)$
$\quad \land \; heartbeat \in HeartbeatStates$
$\quad \land \; process \in ProcessStates$
$\quad \land \; signal \in SignalStates$

$RaiseEvent(event) \triangleq$
$\quad \land Len(events) < MaxEvents$
$\quad \land events' = Append(events, event)$

---

Heartbeat behaviour emulates regular heartbeats and handles timeout cases.

$SendHeartbeatRequest \triangleq$
$\quad \land process = \text{"running"}$
$\quad \land heartbeat = \text{"ready"}$
$\quad \land heartbeat' = \text{"req"}$
$\quad \land \text{UNCHANGED } \langle events, process, signal \rangle$

$SendHeartbeatReply \triangleq$
$\quad \land process = \text{"running"}$    The only case heartbeats come back is
$\quad \land heartbeat = \text{"req"}$    when process is running.
$\quad \land heartbeat' = \text{"ready"}$
$\quad \land \text{UNCHANGED } \langle events, process, signal \rangle$

$TimeoutHeartbeat \triangleq$
$\quad \land heartbeat = \text{"req"}$
$\quad \land heartbeat' = \text{"timeout"}$

$\wedge\ \mathit{RaiseEvent}(\text{"timeout"})$
$\wedge\ \textsc{unchanged}\ \langle \mathit{process}, \mathit{signal} \rangle$

$\mathit{StopHeartbeat} \triangleq \mathit{heartbeat}' = \text{"ready"}$

Stops sending heartbeat messages to the child process.

---

Signal behaviours describe how different modules relate to some of the *UNIX* signals being raised.

$\mathit{OpenSignal} \triangleq$
$\wedge\ \mathit{process} = \text{"running"}$
$\wedge\ \mathit{signal} = \text{"ready"}$
$\wedge\ \mathit{signal}' = \text{"listening"}$
$\wedge\ \textsc{unchanged}\ \langle \mathit{events}, \mathit{heartbeat}, \mathit{process} \rangle$

$\mathit{CloseSignal} \triangleq \mathit{signal}' = \text{"ready"}$

Closing the signal causes the Heartbeat program to cease handling future *UNIX* signals. The signal state "ready" represents signal having been closed.

$\mathit{HandleSignal} \triangleq$
$\wedge\ \mathit{signal} = \text{"listening"}$
$\wedge\ \mathit{RaiseEvent}(\text{"signaled"})$
$\wedge\ \textsc{unchanged}\ \langle \mathit{heartbeat}, \mathit{process}, \mathit{signal} \rangle$

---

Process behaviour advances the state of the inferior process.

$\mathit{StartProcess} \triangleq$
$\wedge\ \mathit{process} = \text{"ready"}$
$\wedge\ \mathit{process}' = \text{"running"}$
$\wedge\ \textsc{unchanged}\ \langle \mathit{events}, \mathit{heartbeat}, \mathit{signal} \rangle$

$\mathit{KillProcess} \triangleq \mathit{process}' = \text{"killed"}$

Represents an operation that kills the process.

$\mathit{TerminateProcess} \triangleq \mathit{process}' = \text{"terminated"}$

Represents an operation that terminates the process.

$\mathit{CompleteProcess} \triangleq$

Represents that the process has completed normally.

$\wedge\ \mathit{process} = \text{"running"}$
$\wedge\ \vee\ \mathit{Len}(\mathit{events}) = 0$
$\quad\ \ \vee\ \neg\{\mathit{events}[n] : n \in 1\,..\,\mathit{Len}(\mathit{events})\} \subseteq \{\text{"complete"}, \text{"aborted"}\}$

Only enabled if the child process is still runnings.

$\wedge\ \mathit{RaiseEvent}(\text{"complete"})$
$\wedge\ \textsc{unchanged}\ \langle \mathit{process}, \mathit{heartbeat}, \mathit{signal} \rangle$

$\mathit{AbortProcess} \triangleq$

Represents that the process has aborted due to an error.

```
    $\land$ $process = \text{"running"}$
    $\land$ $\lor$ $Len(events) = 0$
        $\lor$ $\neg\{events[n] : n \in 1 \,..\, Len(events)\} \subseteq$
            $\{\text{"complete"}, \text{"aborted"}\}$
    $\land$ $RaiseEvent(\text{"aborted"})$
    $\land$ UNCHANGED $\langle process, heartbeat, signal \rangle$
```

Only enabled if the child process is still runnings.

---

Event behaviour consumes each event in the event queue.

```
$ConsumeTimeoutEvent(event) \triangleq$
    $\land$ $event = \text{"timeout"}$
    $\land$ $KillProcess$
    $\land$ $StopHeartbeat$
    $\land$ $CloseSignal$

$ConsumeAbortedEvent(event) \triangleq$
    Consumes an event where the child process aborts due to some error.
    $\land$ $event = \text{"aborted"}$
    $\land$ $KillProcess$
    $\land$ $StopHeartbeat$
    $\land$ $CloseSignal$

$ConsumeCompleteEvent(event) \triangleq$
    Consumes an event that represents the normal completion of the child process.
    $\land$ $event = \text{"complete"}$
    $\land$ $TerminateProcess$
    $\land$ $StopHeartbeat$
    $\land$ $CloseSignal$

$ConsumeSignaledEvent(event) \triangleq$
    A UNIX signal terminates the heartbeat application as well as the child process.
    $\land$ $event = \text{"signaled"}$
    $\land$ $TerminateProcess$
    $\land$ $StopHeartbeat$
    $\land$ $CloseSignal$
```

---

Restart behaviour describes restarting of a killed process.

```
$RestartProcess \triangleq$
    $\land$ $process = \text{"killed"}$
    $\land$ $process' = \text{"ready"}$
    $\land$ $events' = \langle\rangle$
    $\land$ UNCHANGED $\langle heartbeat, signal \rangle$

$GiveUpProcess \triangleq$
```

Frequent process aborts cause the application to give up restarting it.

```
    $\land\ process = \text{“killed”}$
    $\land\ process' = \text{“terminated”}$
    $\land\ events' = \langle\rangle$
    $\land\ \textsc{unchanged}\ \langle heartbeat,\ signal\rangle$
```

---

```
$Init\ \triangleq$
    $\land\ events = \langle\rangle$
    $\land\ heartbeat = \text{“ready”}$
    $\land\ process = \text{“ready”}$
    $\land\ signal = \text{“ready”}$

$HeartbeatBehaviour\ \triangleq$
    $\lor\ SendHeartbeatRequest$
    $\lor\ SendHeartbeatReply$
    $\lor\ TimeoutHeartbeat$

$ProcessBehaviour\ \triangleq$
    $\lor\ StartProcess$
    $\lor\ CompleteProcess$
    $\lor\ AbortProcess$

$SignalBehaviour\ \triangleq$
    $\lor\ OpenSignal$
    $\lor\ HandleSignal$

$EventBehaviour\ \triangleq$
    $\land\ process \notin \{\text{“killed”},\ \text{“terminated”}\}$
    $\land\ events \neq \langle\rangle$
    $\land\ events' = Tail(events)$
    $\land\ \textsc{let}\ \ event\ \triangleq\ Head(events)\textsc{in}$
        $\lor\ \ ConsumeTimeoutEvent(event)$
        $\lor\ \ ConsumeAbortedEvent(event)$
        $\lor\ \ ConsumeCompleteEvent(event)$
        $\lor\ \ ConsumeSignaledEvent(event)$

$RestartBehaviour\ \triangleq$
    $\lor\ RestartProcess$
    $\lor\ GiveUpProcess$

$TerminationBehaviour\ \triangleq$
    $\land\ process = \text{“terminated”}$
    $\land\ \textsc{unchanged}\ \ vars$

$Next\ \triangleq$
    $\lor\ HeartbeatBehaviour$
    $\lor\ ProcessBehaviour$
```

$\quad\lor\ \mathit{SignalBehaviour}$
$\quad\lor\ \mathit{EventBehaviour}$
$\quad\lor\ \mathit{RestartBehaviour}$
$\quad\lor\ \mathit{TerminationBehaviour}$

$\mathit{Spec}\ \triangleq\ \mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}}$

$\mathit{Invariants}\ \triangleq$

Signal task should not start until after the child process launches.

$\quad\land\ \mathit{process} = \text{“ready”} \Rightarrow \mathit{signal} = \text{“ready”}$

complete and aborted cannot both be in the event queue.

$\quad\land\ \text{LET}\ \mathit{queuedEvents}\ \triangleq\ \{\mathit{events}[n] : n \in 1\,..\,\mathit{Len}(\mathit{events})\}\ \text{IN}$
$\qquad\lnot\{\text{“complete”},\ \text{“aborted”}\} \subseteq \mathit{queuedEvents}$

\* Modification History
\* Last modified *Thu Sep* 01 19:45:16 *KST* 2022 by *hcs*
\* Last modified Sun *Apr* 17 19:22:17 *KST* 2022 by *hcs*
\* Created Sat *Apr* 16 17:31:46 *KST* 2022 by *hcs*