Embedded Systems Programming

Kernel Signal Mechanism
(Module 25)

Yann-Hang Lee
Arizona State University
yhlee@asu.edu
(480) 727-7507

Summer 2014
Scheme of Signal Processing

User Mode

- normal program flow
- signal handler
- return code on the stack

Kernel Mode

An event which traps to kernel

- do_signal()
- handle_signal()
- setup_frame()

- system_call()
- sys_sigreturn()
- restore_sigcontext()
Signaling Process

- **Posting** – action taken when event occurs that process needs to be notified of (signal generation)
  - sending a signal – must be done in kernel mode, i.e. `send_signal`, `kill_proc`, etc.

- **Delivery** – action taken when process recognizes arrival of event (signal handling)
  - Before a process resumes execution in user mode, kernel checks for pending non-blocked signals for it. If yes, `do_signal`

- **Catching** – if user-level signal handler is invoked, process is said to catch the signal
  - in kernel, invoke `handle_signal` and `setup_frame`
  - Process first executes signal handler in user mode before resuming “normal” execution.

- **Pending** – signals that have been posted, but not yet delivered
Kernel Data Structures for Signals

sigset_t : array of signals sent to a process

struct sigaction {
    void (*sa_handler)(); /* handler address, or SIG_IGN, or SIG_DFL */
    sigset_t sa_mask;    /* blocked signal list */
    int sa_flags;        /* options e.g., SA_RESTART */
}

struct task_struct { ...... /* for tracking sent, blocked and pending signals */
    struct signal_struct *signal; /* signal descriptor */
    struct sighand_struct *sighand; /* signal handler descriptor */
    sigset_t blocked, real_blocked;
    sigset_t saved_sigmask; /* restored if set_restore_sigmask() */
    struct sigpending pending;
    ......
}
Real-time System Specification

- **Logical correctness requirements:**
  - The computation produces correct outputs.
  - Models of computation to describe inputs and computations
  - Additional requirements on resource, security, reliability, etc.
  - Finite state machine
    - good for control logic and protocols,
    - transition and activity
  - Data flow – modular computations that are triggered by the availability of input data.

- **Temporal correctness requirements:**
  - The computation produces outputs at the right time
  - When the computation can get started and should be completed
## Specification Patterns

<table>
<thead>
<tr>
<th>Category</th>
<th>Pattern</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Duration</strong> (stimuli and responses)</td>
<td>minimum duration</td>
<td>The system has a minimum 'off' period of 120 seconds before it reenters the cranking mode.</td>
</tr>
<tr>
<td></td>
<td>maximum duration</td>
<td>The system can only operate in engine cranking mode for no longer than 10 seconds at one time</td>
</tr>
<tr>
<td><strong>Periodic</strong></td>
<td>bounded recurrence</td>
<td>The ABS controller checks for wheel skidding every 10 milliseconds.</td>
</tr>
<tr>
<td><strong>Real-time order</strong></td>
<td>bounded response</td>
<td>The detection of and response to rapid deceleration must occur within 0.015 seconds.</td>
</tr>
<tr>
<td></td>
<td>bounded invariance</td>
<td>If Error 502 is received, then the braking system is inhibited for 10 seconds.</td>
</tr>
</tbody>
</table>

RT Specification in FSM

- Duration of staying in a state
- Periodic activity in a state
- Bounded response for each transition
- Accumulated delay between multiple transitions

- Hierarchical FSM
  - a state encloses a FSM
  - enter a state → activate a FSM

- Concurrent FSM
  - FSMs run in parallel (active simultaneously)