Operating Systems
- TinyOS
- Real-time POSIX
- Real-time schedulability analysis

Real-Time POSIX
- Standard of UNIX
- Supported by many operating systems
  - Variants of UNIX
  - Linux
  - Many commercial RTOS, e.g., VxWorks
- Windows provides similar services

To Be Covered
- Supervisor mode
- Process management
- Scheduling
- Race condition

Supervisor mode
- On processors with supervisor mode, you can do the following only in supervisor (kernel) mode
  - Execute privileged instructions and access special hardware
  - Set real-time priority
  - Device driver
  - Access to a separate address space (the kernel space)
  - This is the mode in which the operating system usually runs.
  - Provide protective barriers between programs.
  - Prevent applications from corrupting OS data.

Supervisor Mode (2)
- Careful about memory access (e.g., pointers) when
  - programs run in supervisor mode
  - Or processor has no supervisor mode
  - Support supervisor mode?
    - SHARC, ATMEL: No
    - Pentium, ARM: Yes

ARM supervisor mode
- Use SWI instruction to enter supervisor mode, similar to subroutine:
  \[ \text{SWI \ CODE\_1} \]
- Sets PC to 0x08.
- Argument to SWI is passed to supervisor mode code.
- Saves CPSR in SPSR.
Trap
- Trap (software interrupt): an exception generated by an instruction.
  - Ex. enter supervisor mode.
  - Ex. call a service routine
- ARM uses SWI instruction for traps.
- SHARC offers three levels of software interrupts.
  - Called by setting bits in IRPTL register.

Exception
- Exception: internally detected error.
- Exceptions are caused by instruction execution
  - unpredictable
- Built on top of interrupt mechanism.
- Exceptions are usually prioritized and vectorized.

Terms
- Interrupt: generate by external devices
- Exception: generate by CPU due to software errors
  - Ex. div by 0
- Trap: generate by software using instructions (enter supervisor mode: open file, read from network etc.)

Processes in POSIX
- Create a process with fork:
  - parent process keeps executing old program:
  - child process executes new program.

fork()
The fork process creates child:

```
childid = fork();
if (childid == 0) {
  /* child operations */
} else {
  /* parent operations */
}
```

execv()
- Overlays child code:
  childid = fork();
  if (childid == 0) {
    execv("mychild", childargs);
    perror("execv");
    exit(1);
  }
  /* parent operations */
```
Process State
- A process can be in one of three states:
  - executing on the CPU;
  - ready to run;
  - waiting for data.

Process Management
- OS keeps track of:
  - process priorities;
  - scheduling state;
  - process activation record.
- Processes may be created:
  - statically before system starts;
  - dynamically during execution.
- OS controls when contexts switches and what process runs.

Priority-based Scheduling
- Every process has a priority.
- CPU goes to the highest-priority active process
- Categories
  - Fixed vs. dynamic priority
  - Preemptive vs. non-preemptive

Preemptive Fixed-Priority Scheduling Example
- Each process has a fixed priority (1 is the highest):
  - T1: priority 1, execution time 10
  - T2: priority 2, execution time 30
  - T3: priority 3, execution time 20

Preemptive Fixed Priority Scheduling Example (cont.)
- Widely supported by existing RTOS
  - POSIX standard
  - Real-time priorities in POSIX, Linux, Solaris, and Windows
  - Most RTOS: VxWorks...
- Priority is not the only possible mechanism
  - Clock-driven scheduling
  - Reservation-based scheduling
  - Proportional share scheduling
Real-Time Scheduling

- What's the best scheduling algorithm for a workload?
- Can we meet all deadlines?
- How much CPU horsepower do we need to meet our deadlines?

Terminology

- Task
  - usually corresponds to a process or thread
  - may be released multiple times
- Periodic task
  - Ideal: inter-arrival time = period
  - General: inter-arrival time ≥ period
- Aperiodic task
  - Inter-arrival time does not have a lower bound
- Job: an instance of a task

Timing Parameters

- Task Ti
  - Start time
  - Period: p
  - Worst-case execution time: c
  - Relative deadline: d
- Job Jk
  - Release time: time when process becomes ready
  - Finish time
  - Response time r = Finish time - Release time
  - Absolute deadline = Release time + d

Deadline Miss

- A job misses its deadline if
  - response time > relative deadline
  - finish time > absolute deadline
- What happens if a job misses its deadline?
  - Hard deadline: system fails if missed.
  - Soft deadline: user may notice, but system doesn't necessarily fail.

Embedded vs. General-purpose Systems

- General-purpose systems
  - e.g., PCs, database servers
  - Fairness to all tasks (no starvation)
  - Optimize throughput
  - Optimize average performance
- Embedded systems
  - Meet all deadlines
  - Fairness or throughput is not important
  - Hard real-time: worry about worst case performance

Metrics for Scheduling Algorithms

- Ability to satisfy all deadlines.
  - A task set is schedulable under a scheduling algorithm if all jobs can meet their deadlines
  - Run-time overhead: time required for scheduling decision and context switch.
Operating Systems

- TinyOS
- Real-time POSIX
- Real-time schedulability analysis

### Benefit of Scheduling Analysis: Case Study

<table>
<thead>
<tr>
<th>VEST (UVA)</th>
<th>Baseline (Boeing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design - one processor</td>
<td>40</td>
</tr>
<tr>
<td>Scheduling analysis - MUF</td>
<td>1</td>
</tr>
<tr>
<td>Design - two processors</td>
<td>25</td>
</tr>
<tr>
<td>Total composition time</td>
<td>172</td>
</tr>
<tr>
<td>Total composition time</td>
<td>243</td>
</tr>
</tbody>
</table>

- Schedulability analysis reduces composition time by 50%!
- Reduce wasted implementation/testing rounds
- Analysis time <<< testing
- More reduction expected for more complex systems
- Quick exploration of design space!