CENG 448 — Real Time Operating Systems

Modularization

Dr. Larry D. Pyeatt

South Dakota School of Mines and Technology

Modularization

Use an “outside-in” approach.

Look at the I/O devices, and design drivers for them.

You may want to also design a task to manage each device.

Some I/O devices will need more than one task, and some tasks may handle multiple I/O devices.

Device Dependencies

Some devices may operate asynchronously to other devices.

Some devices may have dependencies on other devices.

Mapping out the dependencies can help to determine what tasks are needed.

I/O devices may be *active* or *passive*.

Active:	Generates interrupts. Interrupts could be periodic, or in synch with other devices, or interrupts could be asynchronous.
Passive:	Does not generate interrupts. The application can communicate with the device periodically or aperiodically. Polling must be often enough to not miss data, but seldom enough not to waste CPU time.

Guidelines

- Guideline 1: Identify Device Dependencies
  - Guideline 1a: Identify Active I/O Devices
  - Guideline 1b: Identify Passive I/O Devices
- Guideline 2: Identify Event Dependencies
- Guideline 3: Identify Time Dependencies
  - Guideline 3a: Identify Critical and Urgent Activities
  - Guideline 3b: Identify Different Periodic Execution Rates
  - Guideline 3c: Identify Temporal Cohesion
- Guideline 4: Identify Computationally Bound Activities
- Guideline 5: Identify Functional Cohesion
- Guideline 6: Identify Tasks that Serve Specific Purposes
- Guideline 7: Identify Sequential Cohesion

Some Recommendations Regarding Active Devices

- Assign separate tasks for separate active asynchronous I/O devices.
- Combine tasks for I/O devices that generate infrequent interrupts having long deadlines.
- Assign separate tasks to devices that have different input and output rates.
- Assign higher priorities to tasks associated with interrupt-generating devices.

Some Recommendations Regarding Passive Devices

- Assign a single task to interface with passive I/O devices when communication with them is aperiodic and when deadlines are not urgent.
- Assign separate polling tasks to send periodic requests to passive I/O devices.
- Trigger polling via timer events.
- Assign a high relative priority to polling tasks with relatively short periods.

Event and Time Dependencies

Events may propagate through multiple tasks.

Events can be externally generated (interrupts, polling) and propagate inwards, or internally generated (error conditions) and propagate outwards.

Deadlines cause timing dependencies. After the timing deadlines are determined, separate tasks can be created to handle the deadlines. Some deadlines are *critical*. Others may be *urgent*. Use priorities.

Periodic activities with similar timing requirements can be grouped into tasks.

Temporal cohesion refers to sequences of code that alays execute at the same time, although they are functionally unrelated. Group such activities together.

Computationally Bound Activities

Some tasks may require a lot of CPU time.

These tasks usually have long deadlines.

Run them at low priorities, and use time-slicing.

Functional Cohesion, Single-Purpose Tasks, and Sequential Cohesion

If two tasks are passing a lot of data to each other, then it may be better to combine them into one task.

This will reduce overhead.

Some tasks are single-purpose, and keeping them on their own may help reduce bugs.

Some activities must occur in a specific sequence. In this case, it may be best to combine them into one task.

Schedulability Analysis

Do you have enough CPU power to do what needs to be done?

If not, can you re-organize your tasks?

Rate Monotonic Analysis is a modeling approach that attempts to answer these questions.

$$\frac{C_1}{T_1} + \ldots + \frac{C_n}{T_n} \leq U(n)$$

where C_i is the worst case execution time for task i, T_i is the period for task i, and n is the number of tasks.

U_n is the utilization factor

$$U(n) = n(2^{\frac{1}{n}} - 1)$$

RMA Assumptions

- all tasks are periodic
- tasks are independent (no interactions)
- tasks deadline is the beginning of its next period
- each task has a constant execution time
- all of the tasks have the same level of priority
- aperiodic tasks only do initialization and error recovery, and have no hard deadlines

Extended RMA

Extended Rate Monotonic Analysis accounts for blocking.

$$\frac{C_1}{T_1} + \ldots + \frac{C_i}{T_i} + \frac{B_i}{T_i} \leq U(i), (1 \leq i \leq n)$$

where B_i is the worst case blocking time for task i, C_i is the worst case execution time for task i, T_i is the period for task i, and n is the number of tasks.

U_n is the utilization factor

$$U(i) = i(2^{\frac{1}{i}} - 1)$$