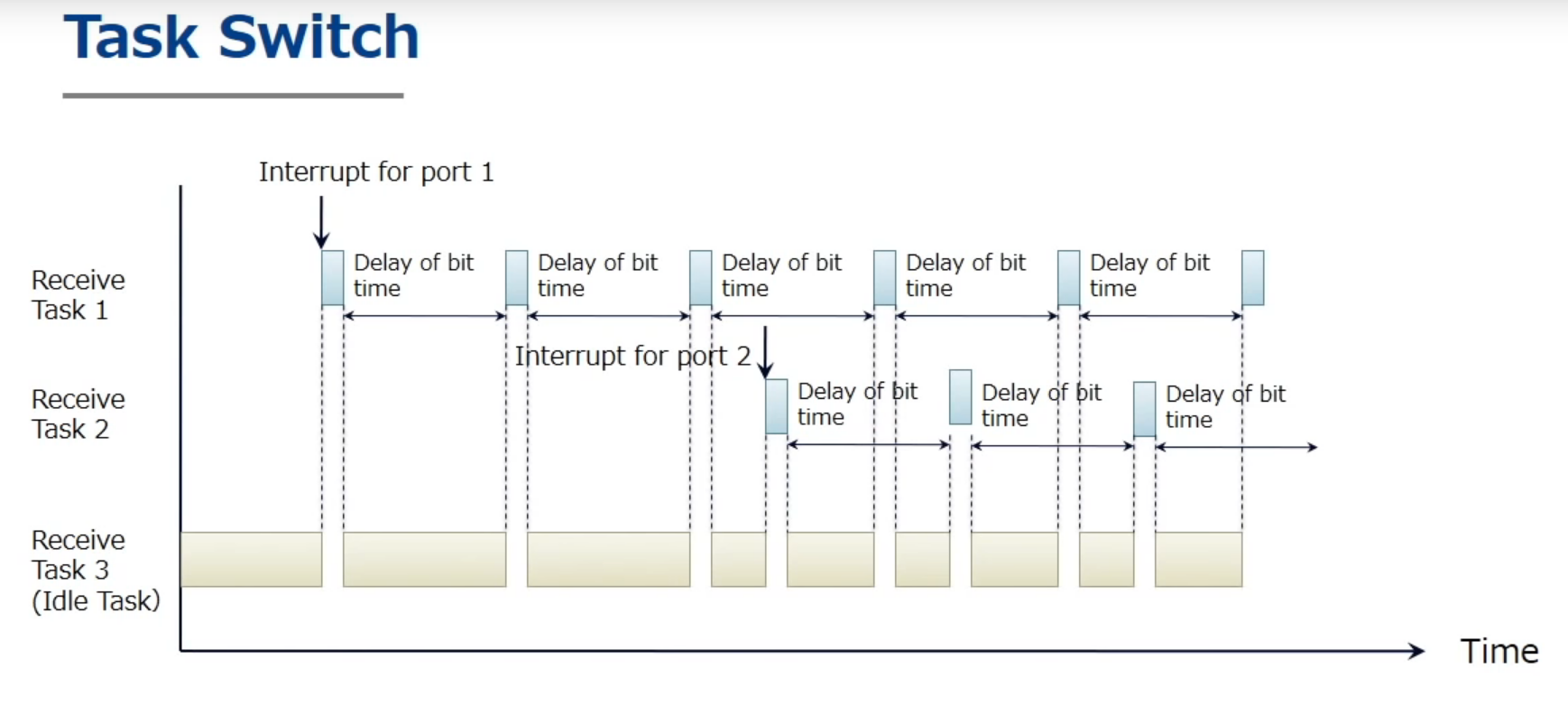
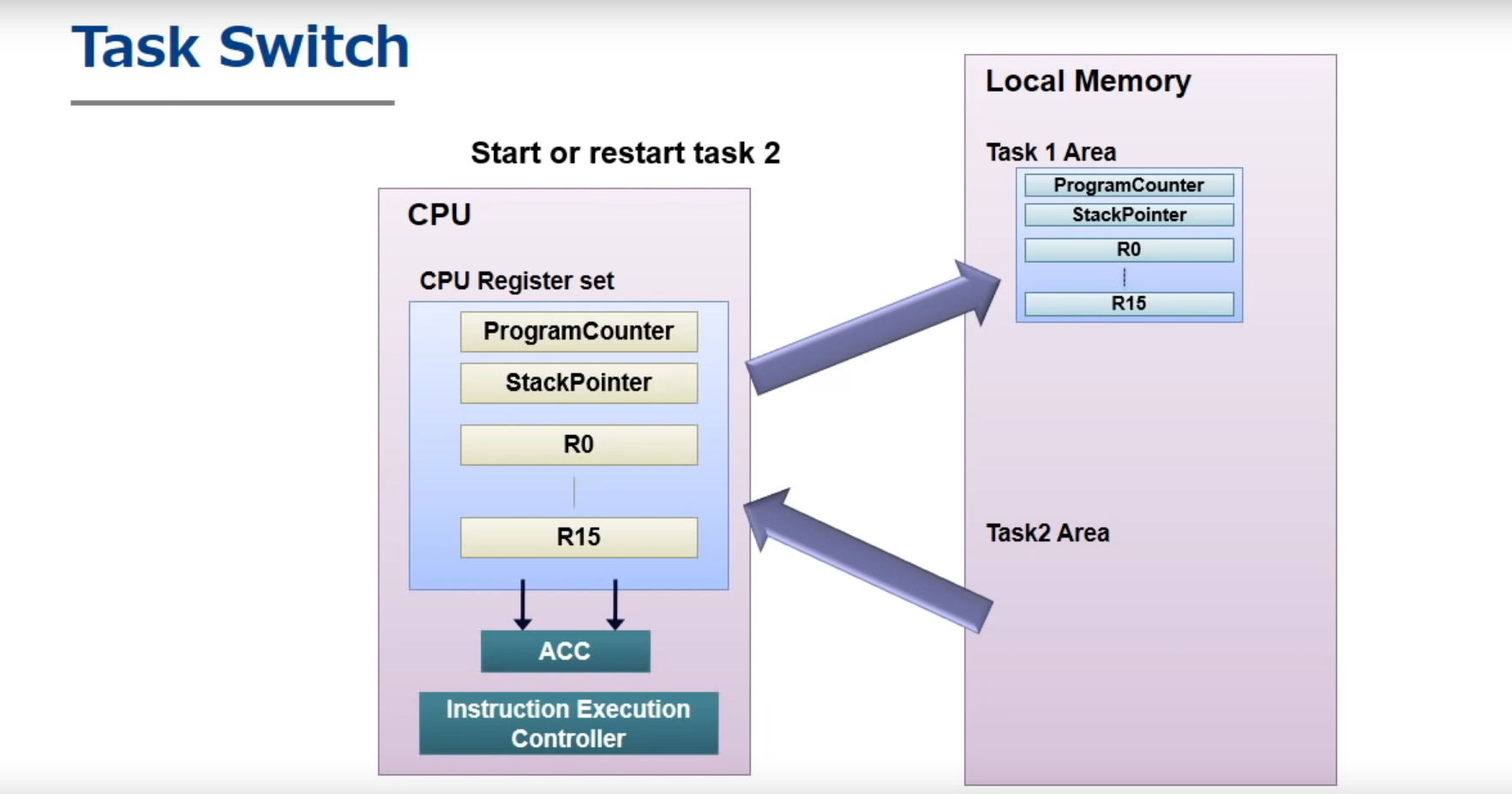
* Week 6
  + Introduction to Realtime Linux
    - <https://www.youtube.com/watch?v=BKkX9WASfpI>
  + RTOS Tutorial (2/5) : Task, handler and API
    - <https://www.youtube.com/watch?v=3mu-2pM04xE&t=201s>
    - A task is a software module handled by the RTOS
      * Tasks are the software application that the RTOS controls
    - The RTOS activates, reactivates (restarts), terminates, and pre-empts tasks
    - Tasks can also use system calls
      * System calls are also referred to as APIs
    - Task switching
      * When a task is idle, the RTOS switches to (activates or reactivates) other tasks as needed
      * 
      * But how exactly do RTOS (real time operating systems) manage this task switching?
        + Let's assume that task 1 is running on the CPU
        + At a given point (if it's time to switch to task 2), then first the current CPU register set (including the program counter and stack pointer) is written to local memory in the task1 save area
        + Then, all of the register values saved in the task2 save area are written to the CPU registers
        + Finally, the CPU starts running
        + It fetches instructions as per (per = in accordance with) the new program counter
        + In other words, it will activate, or reactivate, task 2
        + Basically, task switching is done by replacing data in the CPU registers



* + - More definitions of terms
      * Context
        + Originally context refers to the flow of software processing
        + Deriving from this, context now means the CPU register set contents of a given executed process

As a result, the rewriting of the CPU registers as we just discussed, is called a context switch

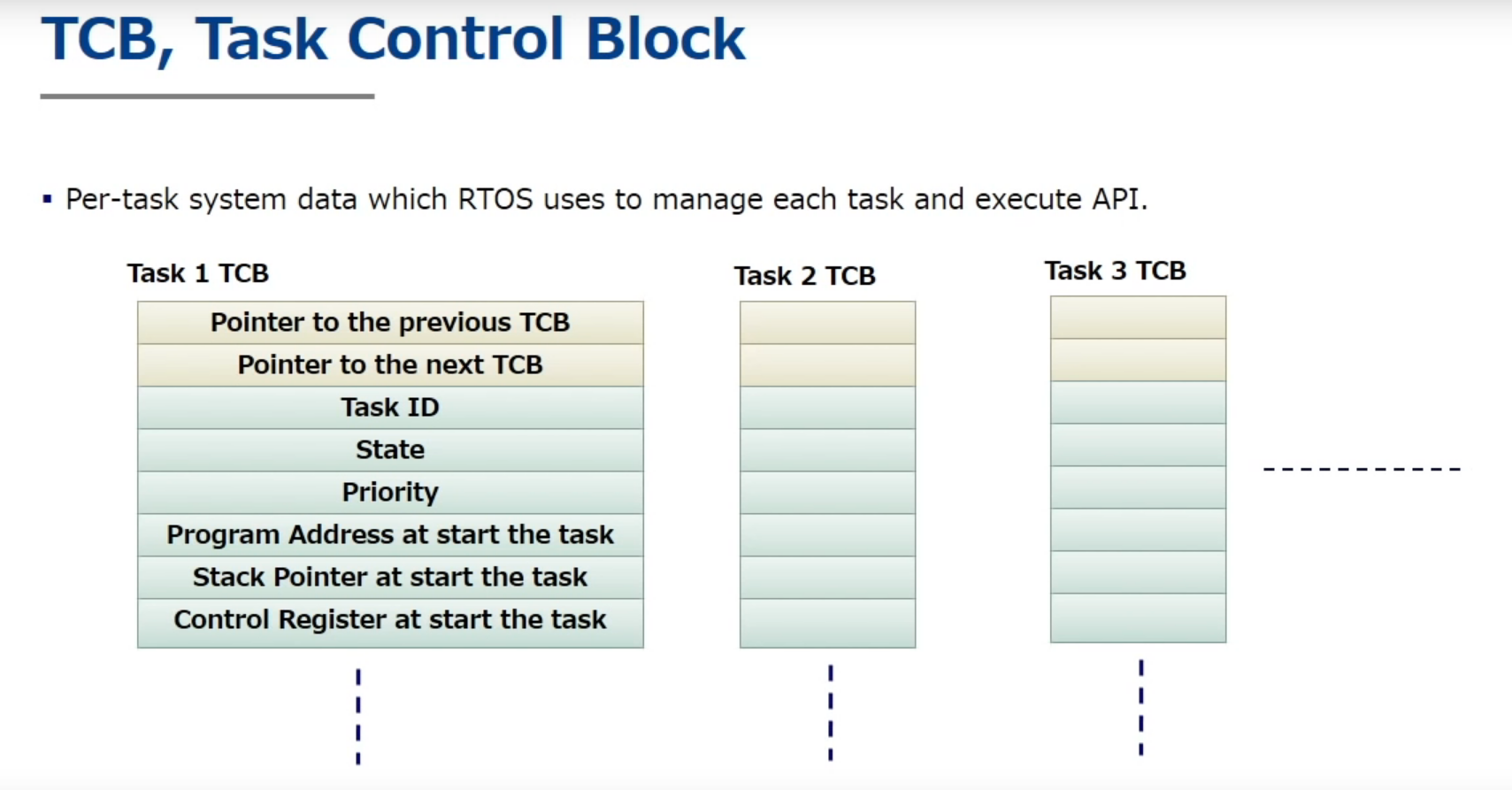
Whenever a context switch occurs, a task switch is executed

* + - * + Dispatch

A dispatch is when one task transfers execution rights to another

Therefore, dispatch processes accompany context switches

* + - TCB (Task Control Block)
      * The TCB (Task Control Block) stores data related to each task
        + The TCB is the per-task system data which RTOS uses to manage each task and execute API
      * The RTOS refers to the TCB whenever it switches tasks or whenever a task invokes an API
        + And whenever necessary, it updates the data in the TCB as it runs processes



* + - Every task is in a state
      * The number of states is dependent on the specific RTOS you are using but every RTOS has at least these 4 states
        + Running

A task in this state means a task is executing software

Any task currently being executed is in running state

Only one task can be executed on the CPU at a time, therefore,

Only one task can be in running state

A task in this state means that it is the highest priority at the time

* + - * + Ready

Indicates that a task can be executed at any time (a task is prepared to execute)

Since another task is already running, a task in the ready state is waiting for that task to transition out of of the running state

The task is waiting to execute because there are some “ready” tasks with higher priority

The task was “running” but was preempted

* + - * + Waiting

To be in waiting means that a task is waiting for a certain event

For example, as discussed earlier, when a receive task invokes the delay API, it goes into waiting

In this case, the awaited event is the completion of the wait time indicated in the argument

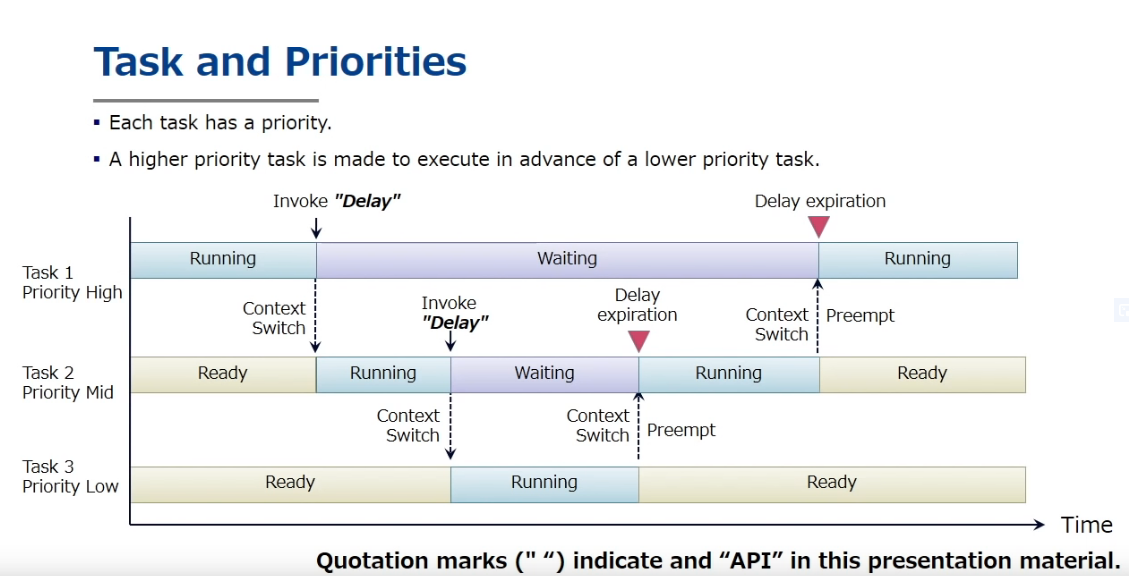
The task has abandoned the right to execute

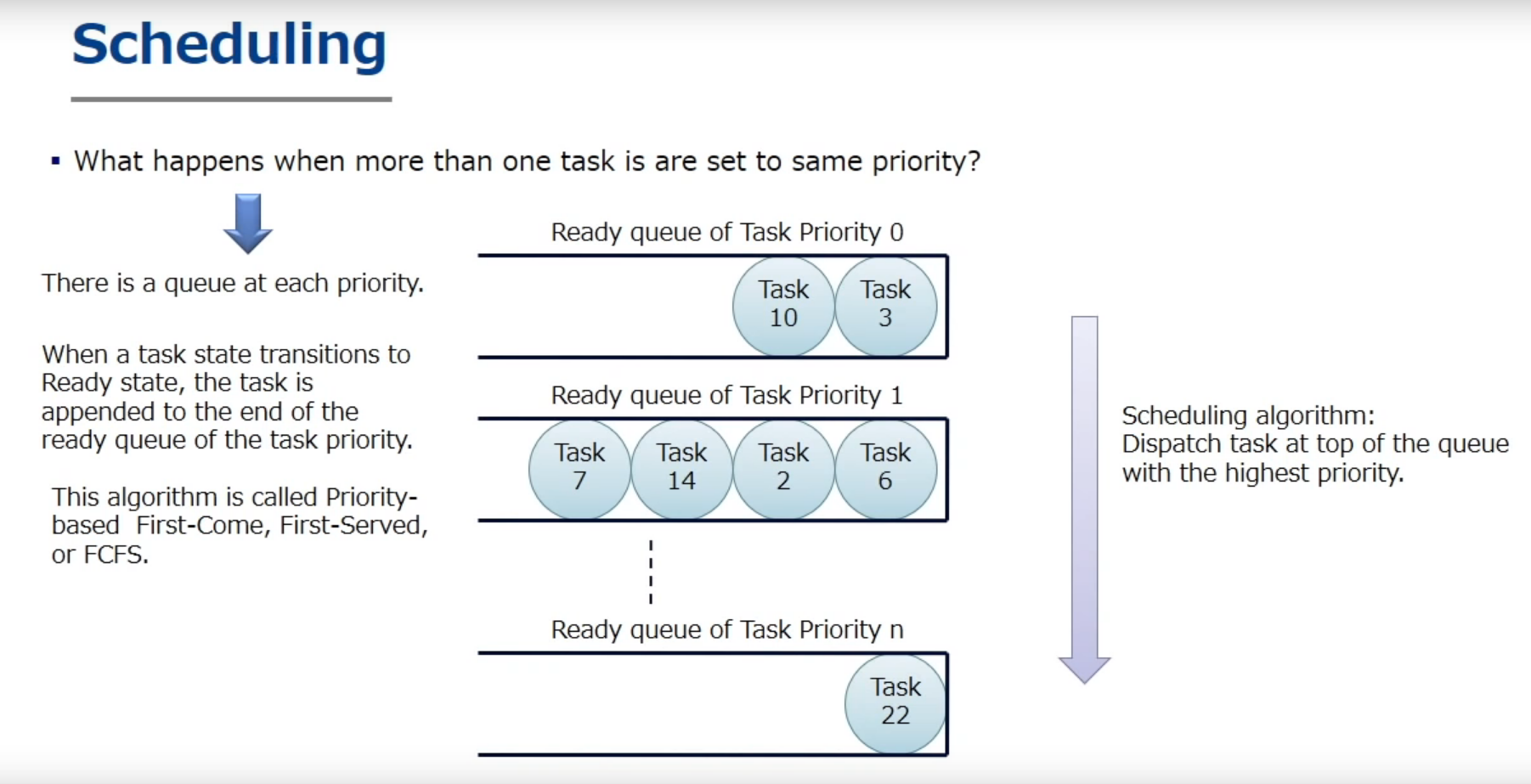
* + - * + Dormant

Dormant state means that a task is registered to the RTOS but it is kept at rest without being activated

A dormant state task is a task which is not working. The task is only registered

* + - * Demonstration
      * Note the following
        + Delay is an API that is invokes by a task



* + - * + When the “delay” API is invoked on a task (task 1) that is running, a context switch is carried out. The task (task 1) goes into a waiting state. Then the RTOS chooses the highest priority task that is in ready state (task 2) to go into running state.
    - Scheduling
      * When multiple tasks have the same priority level they us the FCFS (First come first serve) algorithm
      * 
    - Main takeaways
      * Context switches are when the CPU switches from one task to another
      * A task is an application program managed by the RTOS
      * Each task has an assigned [priority
      * Each task has a state
        + The RTOS provides the services through task state transition
      * Scheduling is to select a task for the running state
        + Most popular scheduling algorithm is Priority based FCFS (first come first serve)
    - Handlers
      * Tasks
        + The software modules (or application software) which are managed by the the RTOS are called tasks
        + But the RTOS also runs handlers
      * A handler is a process executed by pre-empting currently executed software

Again, remember that pre-emption is when we have a context switch from a low priority task to a high priority task

Aka the cpu register set is rewritten

* + - * + A handler is software executed by interruption of currently executed software

Examples

Interrupt handler (ISR), Cyclic Handler, Exception Handler, etc.

* + - * <https://youtu.be/3mu-2pM04xE?t=584>