CENG 448 — Real Time Operating Systems

Tasks

Dr. Larry D. Pyeatt

South Dakota School of Mines and Technology

Tasks

A task is an independent thread of execution that can compete with other concurrent tasks for processor execution time and other resources.

Defining a task

A task has at least three supporting data areas in memory:

- Task control block (TCB),
- Task stack, and
- Task code/data.

Each task has a unique ID/Name, and is *schedulable*.

It is common for an RTOS to start some *system tasks* such as:

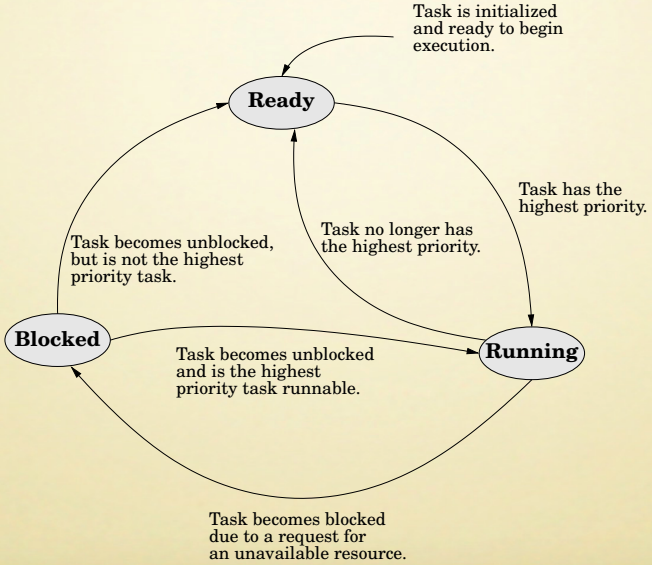
- initilaization/startup task,
- idle task,
- logging task,
- exception-handling task,
- debug agent task,
- etc.

before starting the application tasks.

Idle task

- Runs at lowest priority
- Its job is to
  - perform very low priority housekeeping, and
  - use idle CPU time (or put CPU to sleep),
- Can usually be user-defined, or can call a user function (a hook).

Task states


Task is initialized
and ready to begin
execution.
Ready
Task has the
highest priority.
Task no longer has
the highest priority.
Task becomes unblocked,
but is not the highest
priority task.
Blocked
Running
Task becomes unblocked
and is the highest
priority task runnable.
Task becomes blocked
due to a request for
an unavailable resource.

Kernel task lists

The kernel maintains at least three priority queues: one for each state.

Some kernels may have more states. For example:

- suspended (waiting for another task to wake it up),
- pending (waiting for a specific resource),
- delayed (waiting for a timing delay to end),
- etc.

It *may* maintain separate lists for individual resources.

Typical task management operations

- Creation and Deletion
  - **Create:** Creates a task
  - **Delete:** Deletes a task
- Task Information
  - **Get ID:** Get the task's unique ID.
  - **Get TCB:** Get pointer to task's TCB.
  - **Get Name:** Get task's unique name.

Typical task management operations – continued

- Scheduling

  Suspend: Makes task wait until another task resumes it.

  Resume: Allows task to run again.

  Delay: Puts task to sleep for specified time.

  Restart: Kills and restarts a task.

  Get Priority: Gets task's current priority.

  Set Priority: Sets task's current priority.

  Preemption Lock: Locks out higher priority tasks from preempting the current task.

  Preemption Unlock: Allows higher priority tasks to preempt.

Critical Sections and Priority Inversion

Premption locking allows for *critical sections*.

The ability to get and set priority of running tasks helps avoid *priority inversion*.

Types of task

Run-to-completion tasks will run and exit. Useful for initialization (should be highest priority), or for tasks that run infrequently or only when handling errors.

Continuous tasks include an endless loop, and are intended to continue running as long as the system is powered up. They must include some mechanism to avoid hogging the CPU, by making a *blocking call*. Otherwise, lower priority tasks will never get to run.

Synchronization, Communication, and Concurrency

Tasks synchronize and communicate by using *intertask primitives:*

- semaphores,
- messages (message queues),
- signals,
- pipes,
- etc.

What is the difference between concurrent execution and parallel execution?