UNIT IV
RTOS BASED EMBEDDED SYSTEM DESIGN

4.2 Thread sand Tasks

Thread Concepts

- A thread consists of executable program (codes), *state* of which is controlled by OS,
- The state information– *thread-status* (running, blocked, or finished), *thread structure*— its data, objects and a subset of the process resources, and *thread-stack.* Considered a light weight process and a process level controlled entity.[Light weight means its running does not depend on system resources].

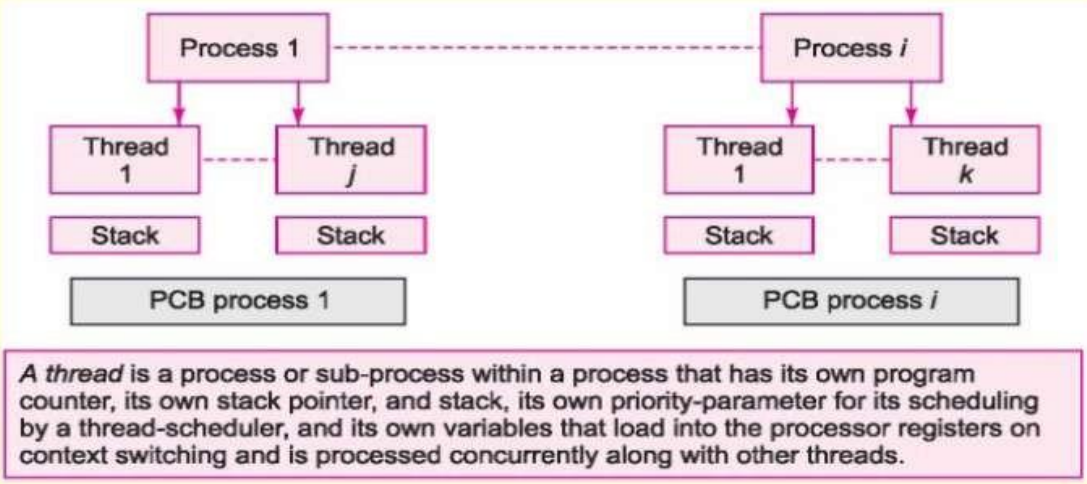
Process…heavyweight

- Process considered as a heavy weight process and a kernel-level controlled entity.
- Process thus can have codes in secondary memory from which the pages can be swapped into the physical primary memory during running of the process.[Heavyweight means its running may depend on system resources]
- May have process structure with the virtual memory map , file descriptors ,user–ID ,etc.
- Can have multiple threads, which share the process structure thread
- *A* process or sub-process within a process that has its own program counter, its own stack pointer and stack, its own priority parameter for its scheduling by a thread scheduler
- Its ‘variables that load into the process or registers on context switching.
- Has own signal mask at the kernel .Thread‘s signal mask
- When un masked lets the thread activate and run.
- When masked , the thread is put into a queue of pending threads.

Thread‘s Stack

- A thread stack is at a memory address block allocated by the OS.

Threads of a Process sharing Process Structure



Application program can be said to consist of number of threads or Processes:

Multi processing OS

- A multi processing OS runs more than one processes.
- When a process consists of multiple threads, it is called multi threaded process.
- A thread can be considered as daughter process.
- A thread defines a minimum unit of a multi threaded process that an OS schedules on to the CPU and allocates other system resources.

Thread parameters

- Each thread has independent parameters ID, priority , program counter, stack pointer, CPU registers and its present status.
- Thread states ─ starting , running ,blocked(sleep) and finished

Thread's stack

- When a function in a thread in OS is called, the calling function state is placed on the stack top.
- When there is return the calling function takes the state information from the stack top
- A data structure having the information using which the OS controls the thread state.
- Stores in protected memory area of the kernel.
- Consists of the information about the thread state

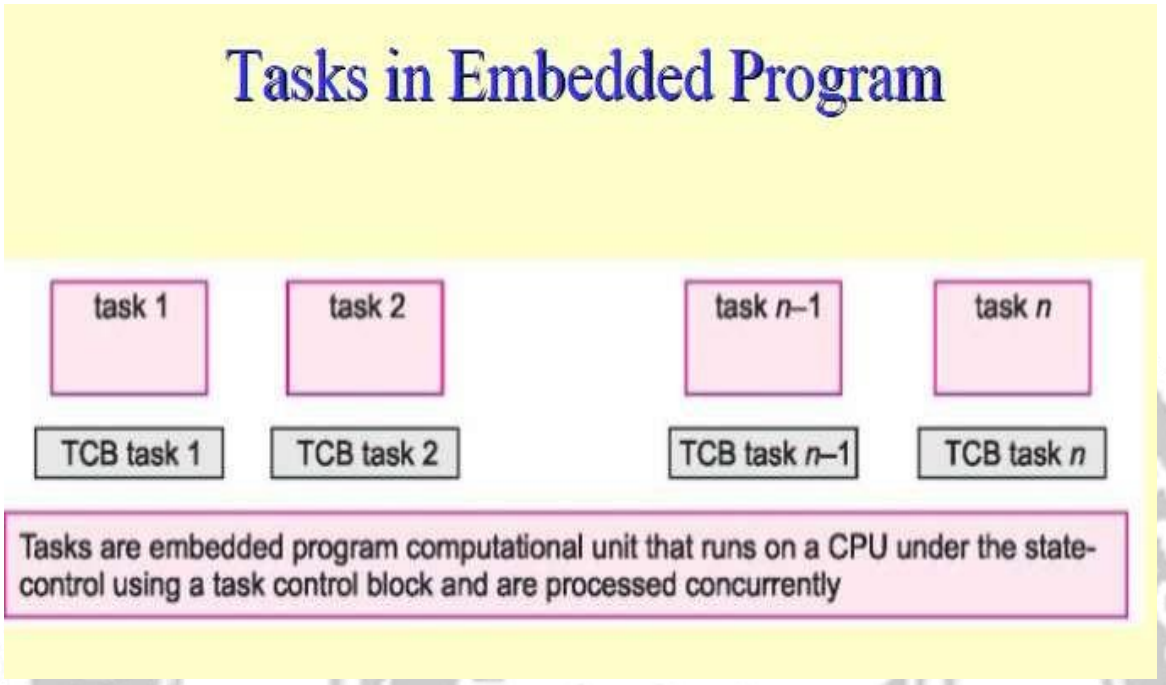
Thread and Task

- Thread is a concept used in Java or Unix.
- A thread can either be a sub-process with in a process or a process within an application program.
- To schedule the multiple processes , there is the concept of forming thread groups and thread libraries.
- A task is a process and the OS does the multitasking.
- Task is a kernel-controlled entity while thread is a process-controlled entity.
- A thread does not call another thread to run. A task also does not directly call another task to run.
- Multi threading needs a thread- scheduler . Multitasking also needs a task-scheduler.
- *There mayor may not be task groups* and task libraries in a given OS

Task and Task States

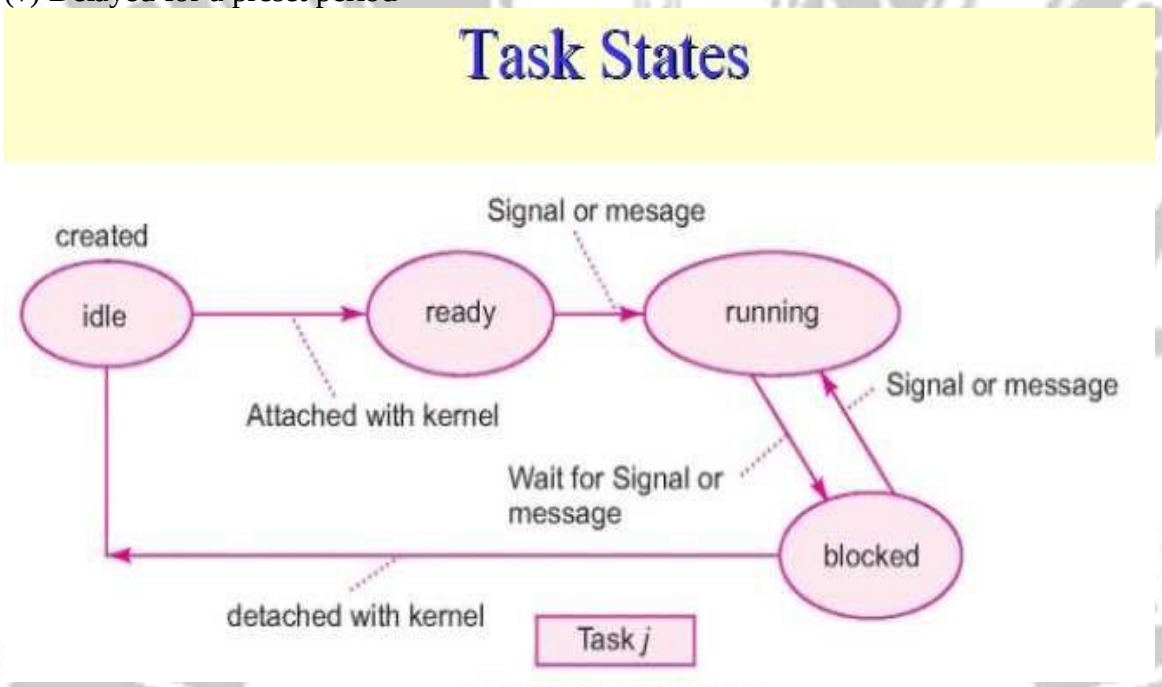
Task Concepts

- An application program can also be said to be a program consisting of the tasks and task behaviors in various states that are controlled by OS.
- A task is like a process or thread in an OS.
- Task─ term used for the process in the RTOS for the embedded systems. For example, V x Works and µ COS-II are the RTOS, which use the term task.
- A task consists of executable program (codes) , *state* of which is controlled by OS , the *state* during running of a task represented by information of process status (running, blocked, or finished) ,process – structure — its data, objects and resources and task control block(PCB).
- Runs when it is scheduled to run by the OS (kernel), which gives the control of the CPU on a task request (system call) or a message.
- Runs by executing the instructions and the continuous changes of its state takes place as the program counter (PC) changes.
- Task is that executing unit of computation , which is controlled by some process at the OS scheduling mechanism , which lets it execute on the CPU and by some process at OS for a resource-management mechanism that lets it use the system memory and other system-resources such as network, file ,display or printer.
- A task─ an independent process.
- No task can call another task. [It is unlike a C (orC++) function , which can call another function.]
- The task ─ can send signal (s) or message (s) that can let another task run.
- The OS can only block a running task and let another task gain access of CPU to run the servicing codes



Task States

(i) Idle state [Not attached or not registered]
(ii) Ready State [Attached or registered]
(iii) Running state
(iv) Blocked(waiting) state
(v) Delayed for a preset period



Idle (created) state

- The task has been created and memory allotted to its structure however, it is not ready and is not schedulable by kernel.

Ready (Active) State

- The created task is ready and is schedulable by the kernel but not running at present as another higher priority task is scheduled to run and gets the system resources at this instance.

Running state

- Executing the codes and getting the system resources at this instance. It will run till it needs some IPC (input) or wait for an event or till it gets pre-empted by another higher priority task than this one.
-

Blocked (waiting) state

- Execution of task codes suspends after saving the needed parameters in to its Context .It needs some IPC (input) or it needs to wait for an event or wait for higher

Priority task to block to enable running after blocking.

Deleted (finished) state

- Deleted Task─ The created task has memory deallotted to its structure . It frees the memory. Task has to be re-created.

Function

- Function is an entity used in any program function task or thread for performing specific set of actions when called and on finishing the action the control returns to the function calling entity (a calling function or task or process or thread).
- Each function has an ID (name)
- has program counter and
- has its stack, which saves when it calls another function and the stack restores on return to the caller.
- Functions can be nested. One function call another, that can call another and so on and later the return is in reverse order