Inter-Process Communication and Synchronization of Processes, Threads and Tasks:

Lesson-3: Task and Task States

Task Concepts

Task

- 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…

- Task─ term used for the process in the RTOSes for the embedded systems.
- For example, VxWorks and µCOS-II are the RTOSes, which use the term task.

<u>Task </u>…

- 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).

Task….

- 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 …

- 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

Task…

- A task— an independent process.
- No task can call another task. [It is unlike a C (or C++) function, which can call another function.]

Task…

- 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

Application program can be said to consist of number of tasks

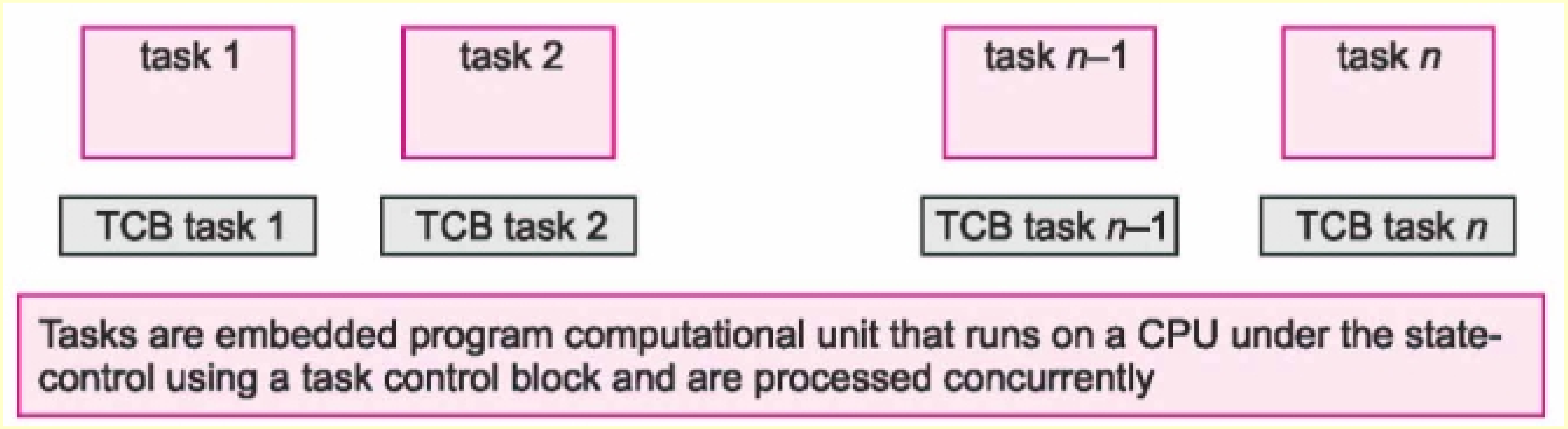
Example – Automatic Chocolate Vending Machine

- Software highly complex.
- RTOS schedules to run the application embedded software as consisting of the number of Tasks

Example – Automatic Chocolate Vending Machine

- Number of functions, ISRs, interrupt-service-threads, tasks, multiple physical and virtual device drivers, and several program objects that must be concurrently processed on a single processor

Tasks in Embedded Program



Tasks are embedded program computational unit that runs on a CPU under the state-control using a task control block and are processed concurrently

Exemplary tasks at the ACVM

- *Task User Keypad Input* — keypad task to get the user input
- *Task Read-Amount* — for reading the inserted coins amount,
- *Chocolate delivery task* — delivers the chocolate and signals the machine for readying for next input of the coins,

Exemplary tasks at the ACVM

- *Display Task*,
- *GUI_Task* —for graphic user interfaces,
- *Communication task* — for provisioning the AVCM owner access the machine Information and information.

Task States

States of a Task in a system

(i) Idle state [Not attached or not registered]

(ii) Ready State [Attached or registered]

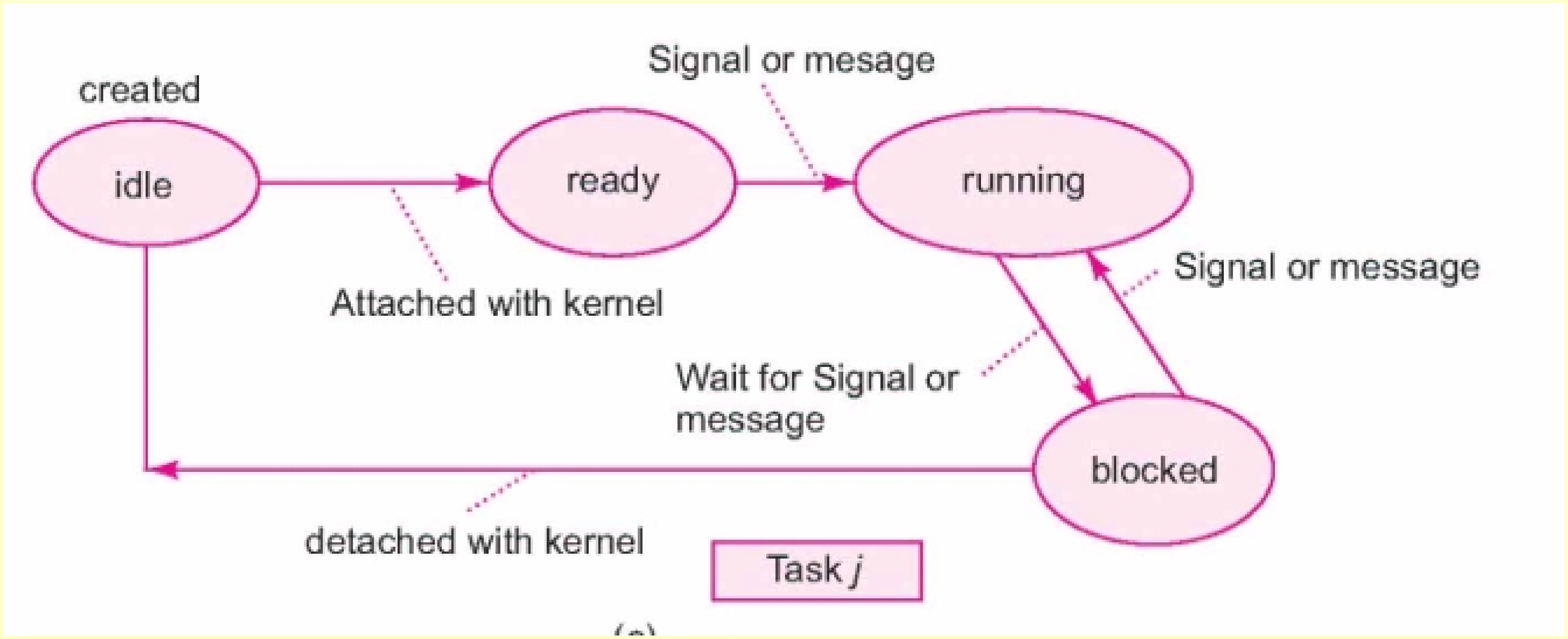
(iii) Running state

(iv) Blocked (waiting) state

(v) Delayed for a preset period

Number of possible states depends on the RTOS.

Task States


created
idle
Attached with kernel
ready
Signal or mesage
running
Signal or message
Wait for Signal or message
blocked
detached with kernel
Task j

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 preempted by another higher priority task than this one.

Blocked (waiting) state

- Execution of task codes suspends after saving the needed parameters into 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.

Blocked (waiting) state example

- A task is pending while it waits for an input from the keyboard or a file. The scheduler then puts it in the blocked state

Deleted (finished) state

Deleted Task—

- The created task has memory de-allotted to its structure.
- It frees the memory.
- Task has to be re-created.

Created and Activated Task States During Processing

one of three states—

- ready,
- running and
- blocked.

OS Functions for the tasks and Task states at Smart Card

Exemplary Steps

- Let the main program first run an RTOS function *OS_initiate* ( )
- This enables use of the RTOS functions
- The main program runs an RTOS function OS_Task_Create ( ) to create a task, Task_Send_Card_Info..

OS Functions

- OS_Task_Create ( ) runs twice to create two other tasks, Task_Send_Port_Output and Task_Read_Port_Input and both of them are also in idle state. Let these tasks be of middle and low priorities, respectively.

OS Functions

- OS_Start ( )─ for starting and
- OS_Ticks_Per_Sec ()─ for initiating *n* system clock interrupts per s.
- After initiation, the system switches from user mode to supervisory mode every 1/60 s if n = 60. All three task-states will be made in ready_state by an OS function

Task_Send_Card_Info

- Task is for sending card information to the host.
- Has an allocated memory for the stack.
- Has a TCB using which OS controls the task.
- Task state is idle state at the brginning.
- Let Task_Send_Card_Info be of high priority

Task_Send_Card_Info

- The OS runs a function, which makes the Task_Send_Card_Info state as running.
- Task_Send_Card_Info runs an OS function mailbox_post (authentication_request), which sends the server identification request through IO port to the host using the task Task_Send_Port_Output

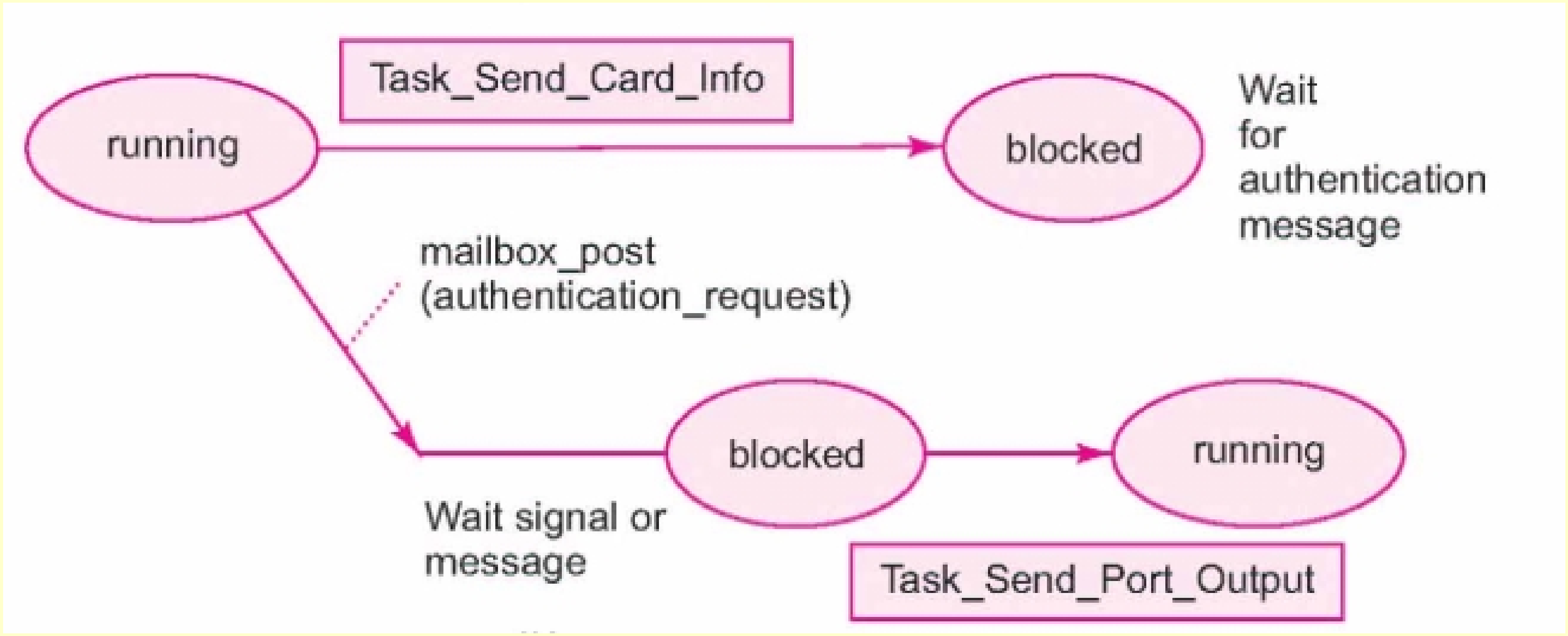
Task_Send_Card_Info

- The Task_Send_Card_Info runs a function mailbox_wait ( ), which makes the task state as blocked and OS switches context to another task Task_Send_Port_Output and then to Task_Read_Port_Input for reading the IO port input data,

Task_Send_Card_Info

- The OS when mailbox gets the authentication message from server, switches context to Task_Send_Card_Info and the task becomes in running state again

Task_Send_Card_Info States



Summary

We learnt

- Application program can be said to consist of number of tasks
- Task defined as a executing computational unit that processes on a CPU and state of which is under the control of kernel of an operating system.

We learnt

- Task state at an instance defines by *task-Information* (running, blocked, or finished), *task-structure*—its data, objects and resources and *task- control block*.

We learnt

- OS lets a task execute on the CPU
- Some process at OS for a resource-management mechanism lets the task use the system-memory and other system-resources such as network, file, display or printer

End of Lesson 3 of Chapter 7