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

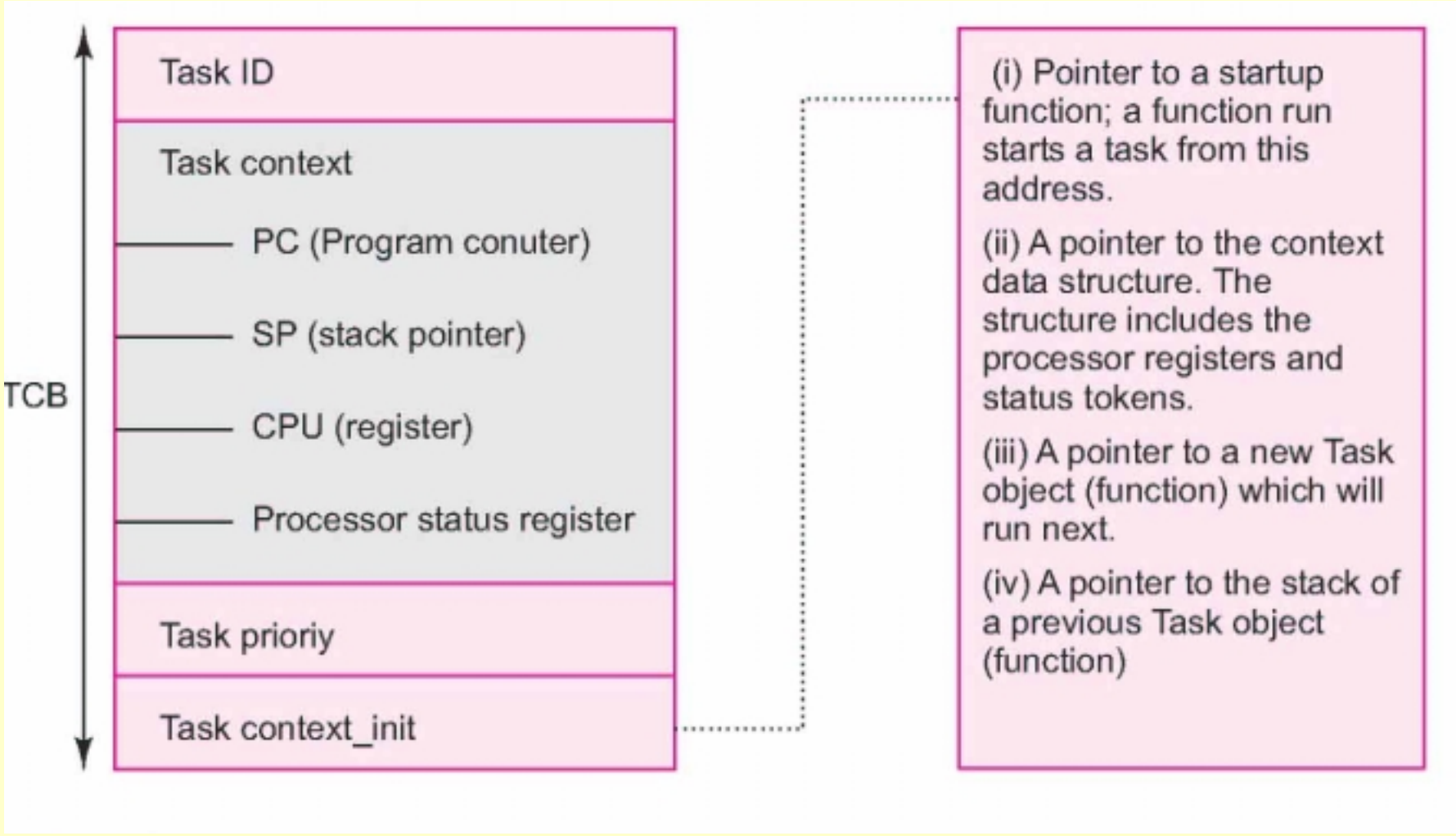
Lesson-4: Task Data, TCB and Characteristics

Task and Data, and Task-Control-Block

Task and its data

- Includes task context and TCB
- TCB– A data structure having the information using which the OS controls the process state.
- Stores in protected memory area of the kernel.
- Consists of the information about the task state

Task and its data including Context



Task Information at the TCB…

- TaskID, for example, ID a number between 0 and 255
- task priority, if between 0 and 255, is represented by a byte
- parent task (if any),
- child task (if any),
- address to the next task's TCB of task that will run next,

Task Information at the TCB…

- allocated program memory address blocks in physical memory and in secondary (virtual) memory for the tasks-codes,
- allocated task-specific data address-blocks
- allocated task-heap (data generated during the program run) addresses,

Task Information at the TCB…

- allocated task-stack addresses for the functions called during running of the process,
- allocated addresses of CPU register-save area as a task context represents by CPU registers, which include the program counter and stack pointer

Task Information at Task Control Block…

- allocated addresses of CPU register-save area as a task context,

[Register-contents (define process context) include the program counter and stack pointer contents]

Information about the process state at Process Control Block…

- task-state signal mask [when mask is set to 0 (active) the process is inhibited from running and when reset to 1, the process is allowed to run],
- Task signals (messages) dispatch table [task IPC functions],

Information about the process state at Process Control Block…

- OS allocated resources' descriptors (for example, file descriptors for open files, device descriptors for open (accessible) devices, device-buffer addresses and status, socket-descriptor for open socket), and
- Security restrictions and permissions

Task's Context and Context switching

Context

- Each task has a context,
- context has a record that reflects the CPU state just before OS blocks one task and initiates another task into running state.
- Continuously updates during the running of a task,
- Saved before switching occurs to another task

Context

- The present CPU registers, which include program counter and stack pointer are part of the context
- When context saves on the TCB pointed process-stack and register-save area addresses, then the running process stops.
- Other task context now loads and that task runs─ which means that the context has switched

Task Coding in Endless Event-Waiting Loop

Task endless event-waiting loop

- Each task may be coded such that it is in endless event-waiting loop to start with.
- An event loop is one that keeps on waiting for an event to occur. On the start-event, the loop starts executing instruction from the next instruction of the waiting function in the loop.

Task endless event-waiting loop…

- Execution of service codes (or setting a token that is an event for another task) then occurs.
- At the end, the task returns to the event waiting in the loop

ACVM Chocolate delivery task

- *static void* Task_Deliver *(void *taskPointer)* {
- /* The initial assignments of the variables and pre-infinite loop statements that execute once only*/

ACVM Chocolate delivery task infinite loop

- while (1) { /* Start an infinite while-loop. */
- /* Wait for an event indicated by an IPC from *Task Read-Amount* */
- .
- /* Codes for delivering a chocolate into a bowl. */
- .
- /* Send message through an IPC for displaying "*Collect the nice chocolate.* Thank you, visit again" to the Display Task*/

ACVM Chocolate delivery task infinite loop

- /* Resume delayed *Task Read-Amount* */
- *};* /* End of while loop*/
- / * End of the Task_Deliver function */

Task's Characteristics

Task Characteristics

- Each task is independent and takes control of the CPU when scheduled by a scheduler at an OS. The scheduler controls and runs the tasks.
- No task can call another task. [It is unlike a C (or C++) function, which can call another function.]

Task Characteristics…

- Each task is recognised by a TCB.
- Each task has an ID just as each function has a name. The ID, *task ID* is a byte if it is between 0 and 255. ID is also an index of the task.

Task Characteristics…

- Each task may have a priority parameter.
- The priority, if between 0 and 255, is represented by a byte.
- Signal mask

Task Characteristics…

- A task is an independent process. The OS will only block a running task and let another task gain access of CPU to run the servicing codes.

Task Characteristics…

- Each task has its independent (distinct from other tasks) values of the followings at an instant: (*i*) program counter and (*ii*) task stack pointer (memory address from where it gets the saved parameters after the scheduler granting access of the CPU). These two values are the part of its context of a task.

Task Characteristics…

- Task runs by context switching to that task by the OS scheduler.
- Multitasking operations are by context switching between the various tasks

Task Characteristics…

- Each task must be either reentrant routine or
- Must have a way to solve the shared data problem

Task Characteristics…

- The task returns to the either idle state (on deregistering or detaching) or ready state after finishing (completion of the running state), that is, when all the servicing codes have been executed.

Task Characteristics…

- Each task may be coded such that it is in endless loop waiting for an event to start running of he codes.
- Event can be a message in a queue or in mailbox or
- Event can be a token or signal
- Event can be delay-period over

Summary

We learnt

- Application program can be said to consist of number of tasks
- Task defined as an 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- state information (running, blocked, or finished), task-structure—its data, objects and resources and task-control block.

We learnt

- TCB– a data structure having the information using which the OS controls the process state
- TCB consists of the information about the process state
- TCB stores in protected memory area of the kernel

We learnt

- Task has a unique ID. It has states in the system as follows: idle, ready, running, blocked, delayed and deleted.
- It is in ready state again after finish when it has infinite waiting loop

End of Lesson 4 of Chapter 7