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

Lesson-2: Thread

# Thread Concepts

### **Thread**

- 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

#### Thread... lightweight

• Considered a lightweight process and a process level controlled entity.

[Light weight means its running does not depend on system resources]

#### Process... heavyweight

- Process considered as a heavyweight 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.

  [Heavy weight means its running may depend on system resources]

#### Process ...

- 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 processor registers on context switching.
- Has own signal mask at the kernel.

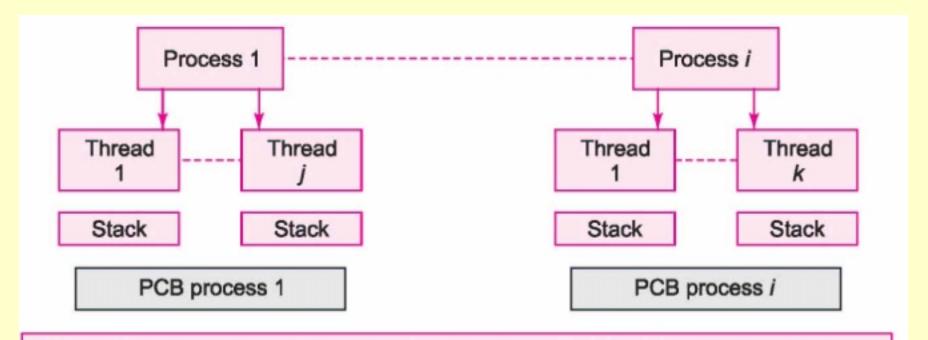
## Thread's signal mask

- When unmasked 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



A thread is 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, and its own variables that load into the processor registers on context switching and is processed concurrently along with other threads.

# Application program can be said to consist of number of threads or processes

#### Multiprocessing OS

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

# Example — Multiple threads of Display process in Mobile Phone Device

- Display\_Time\_Date thread for displaying clock time and date.
- Display\_Battery thread for displaying battery power.
- Display\_Signal thread for displaying signal power for communication with mobile service provider.

# Exemplary threads of display\_process at the phone device

- Display\_Profile thread for displaying silent or sound-active mode. A thread
- Display\_Message thread for displaying unread message in the inbox.
- Display\_Call Status thread —for displaying call status; whether dialing or call waiting

#### Exemplary processes at the phone device

- Display\_Menu thread for displaying menu.
- Display threads can share the common memory blocks and resources allocated to the Display Process.

## Minimum computational unit

 A display thread is now the minimum computational unit controlled by the OS.

### Thread Parameters and Stack

### 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

#### Thread Stack

- 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 and Task

- Thread is a concept used in Java or Unix.
- A thread can either be a sub-process within 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.

#### Thread and task

- A task is a process and the OS does the multitasking.
- Task is a kernel-controlled entity while thread is a process-controlled entity.

### Thread and Task analogy

- A thread does not call another thread to run.
   A task also does not directly call another task to run.
- Multithreading needs a thread-scheduler.
   Multitasking also needs a task-scheduler.
- There may or may not be task groups and task libraries in a given OS

# Summary

#### We learnt

• A thread is 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.

#### We learnt

- Thread is a concept in Java and Unix
- Thread is a lightweight sub-process or process in an application program.
- Thread can share a process structure.
- Thread has thread stack, at the memory.
- It has a unique ID.
- States of thread—starting, running, sleeping (blocked) and finished.

# End of Lesson 2 of Chapter 7