OPERATING SYSTEMS (THEORY)

LECTURE - 4

K.ARIVUSELVAN

Assistant Professor (Senior) – (SITE)

VIT University

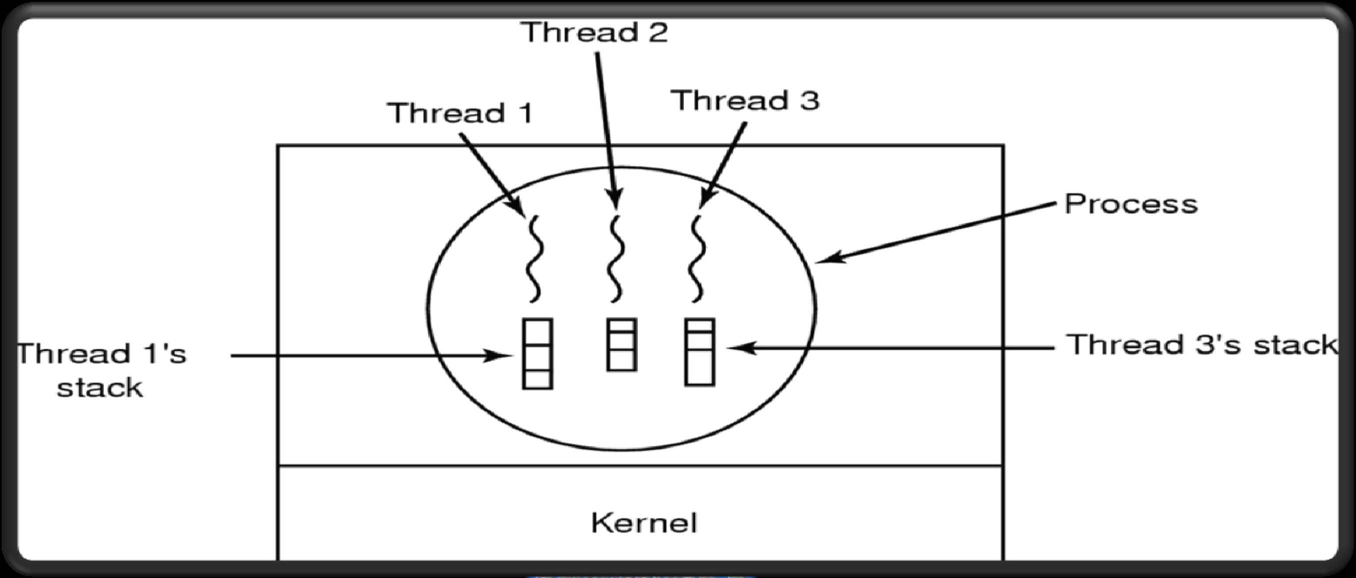
THREADS

What is a Thread?

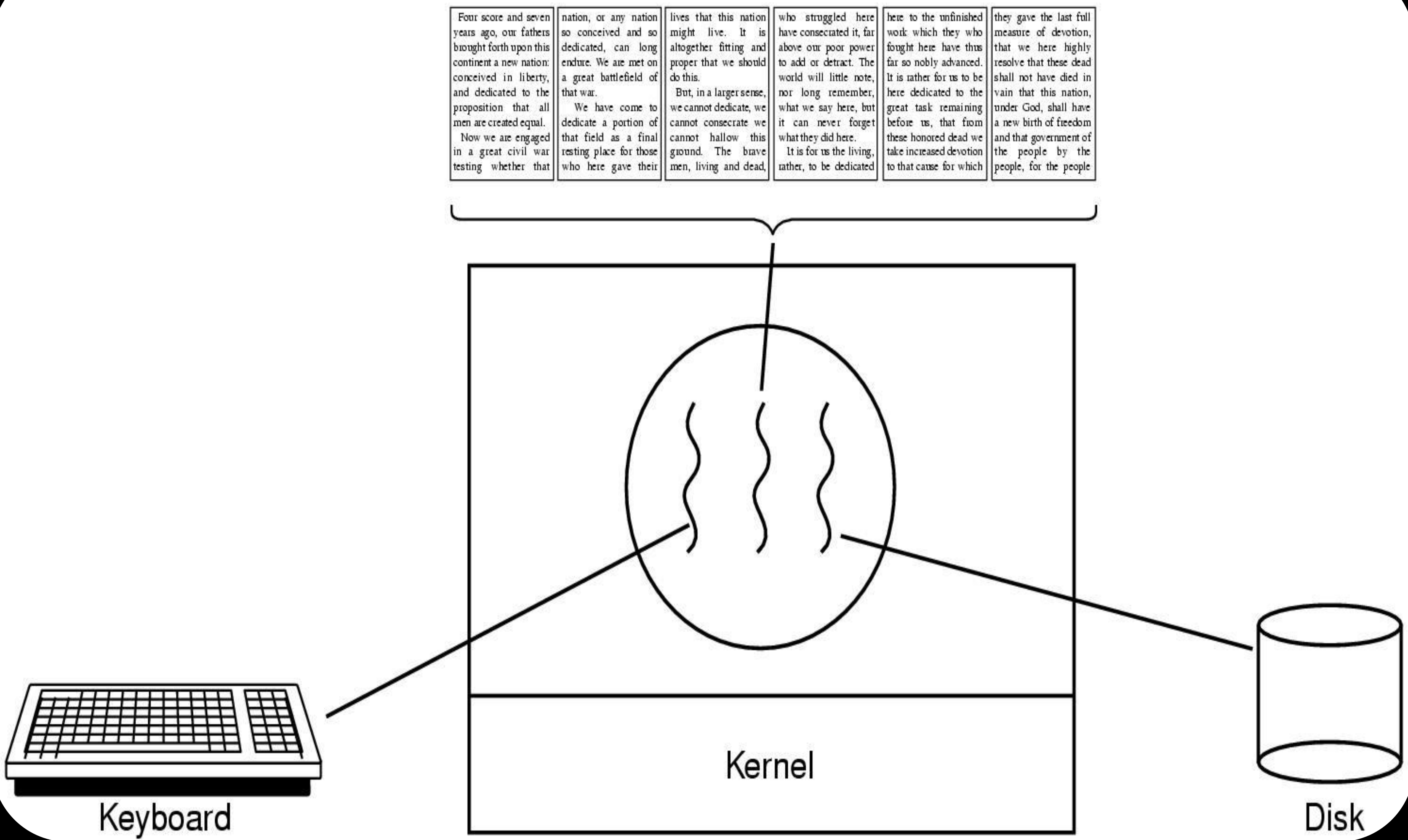
A **process** is divided into number of **light weight process**, Each **lightweight process** is called as **thread**

Each thread has its own:

- Program Counter
- Registers (holds current working variables)
- Stack (Contains execution history)
- Thread State



Example (Word Processor)



Four score and seven years ago, our fathers brought forth upon this continent a new nation: conceived in liberty, and dedicated to the proposition that all men are created equal.

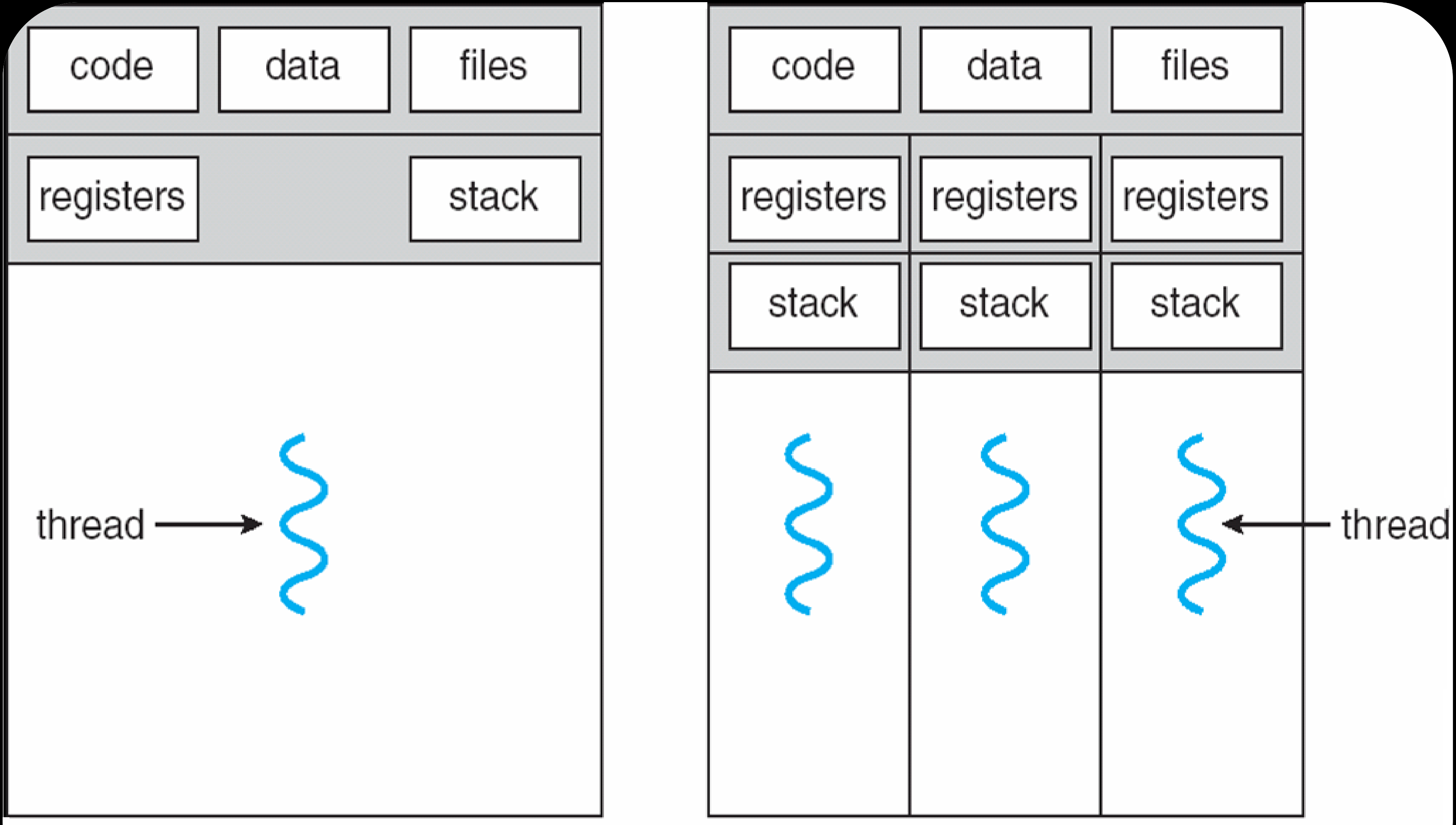
Now we are engaged in a great civil war testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battlefield of that war.

We have come to dedicate a portion of that field as a final resting place for those who here gave their lives that this nation might live. It is altogether fitting and proper that we should do this.

But, in a larger sense, we cannot dedicate, we cannot consecrate we cannot hallow this ground. The brave men, living and dead, who struggled here have consecrated it, far above our poor power to add or detract. The world will little note, nor long remember, what we say here, but it can never forget what they did here.

It is for us the living, rather, to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us, that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion, that we here highly resolve that these dead shall not have died in vain that this nation, under God, shall have a new birth of freedom and that government of the people by the people, for the people

Single and Multithreaded Processes



single-threaded process

multithreaded process

PROPERTIES

- Threads can share

=> Address space

=> Opened Files & Other resources

=> CPU

(but only one thread is active at a time)

- Threads can create

=> Childs threads

- Threads are not independent of one another

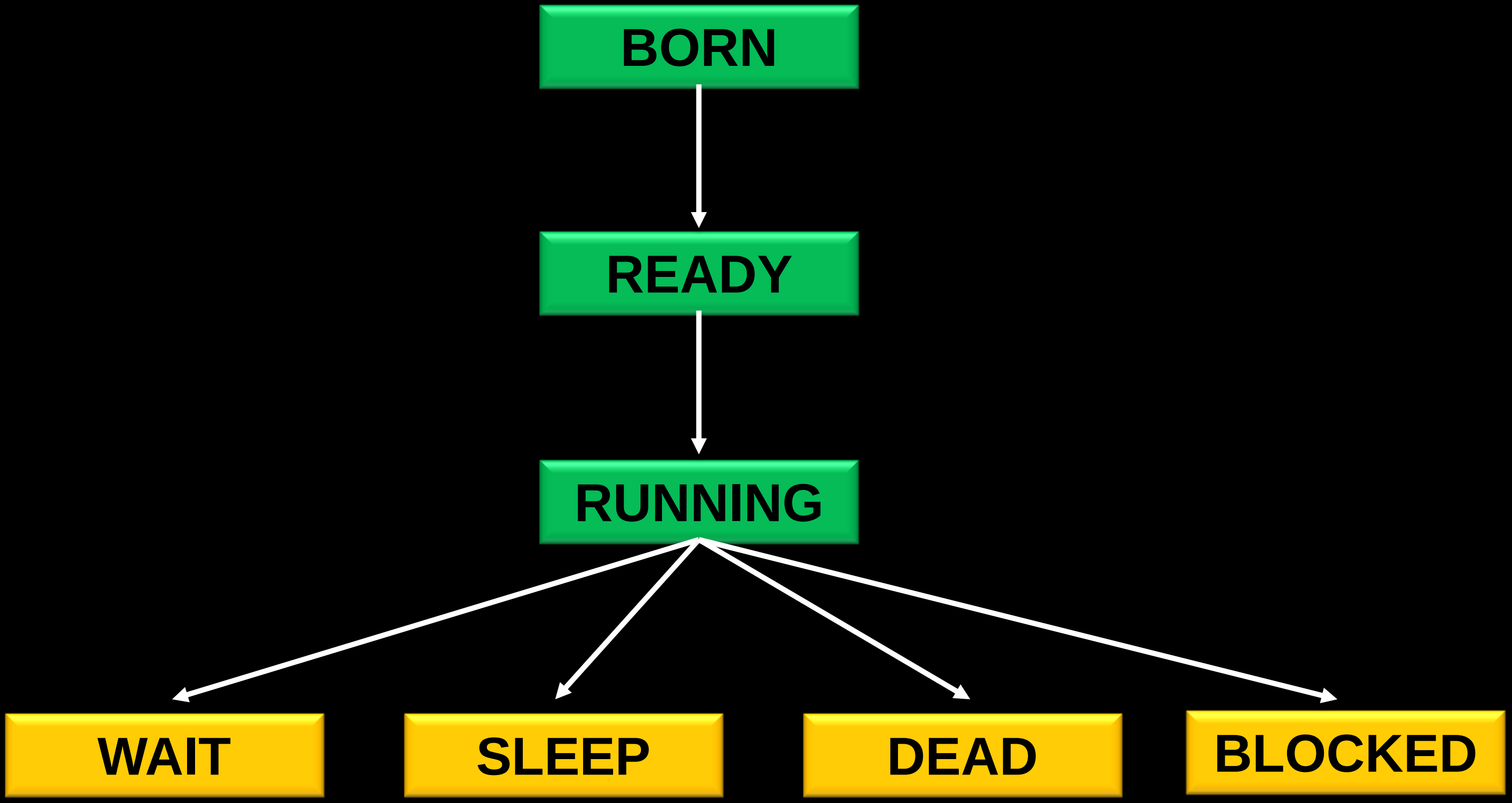
BENEFITS

- Less time to create and terminate a thread than a process (because we do not need another address space).

- Less time to switch between two threads than between processes.

- Inter-thread communication and synchronization is very fast

THREAD LIFE CYCLE



BORN

READY

RUNNING

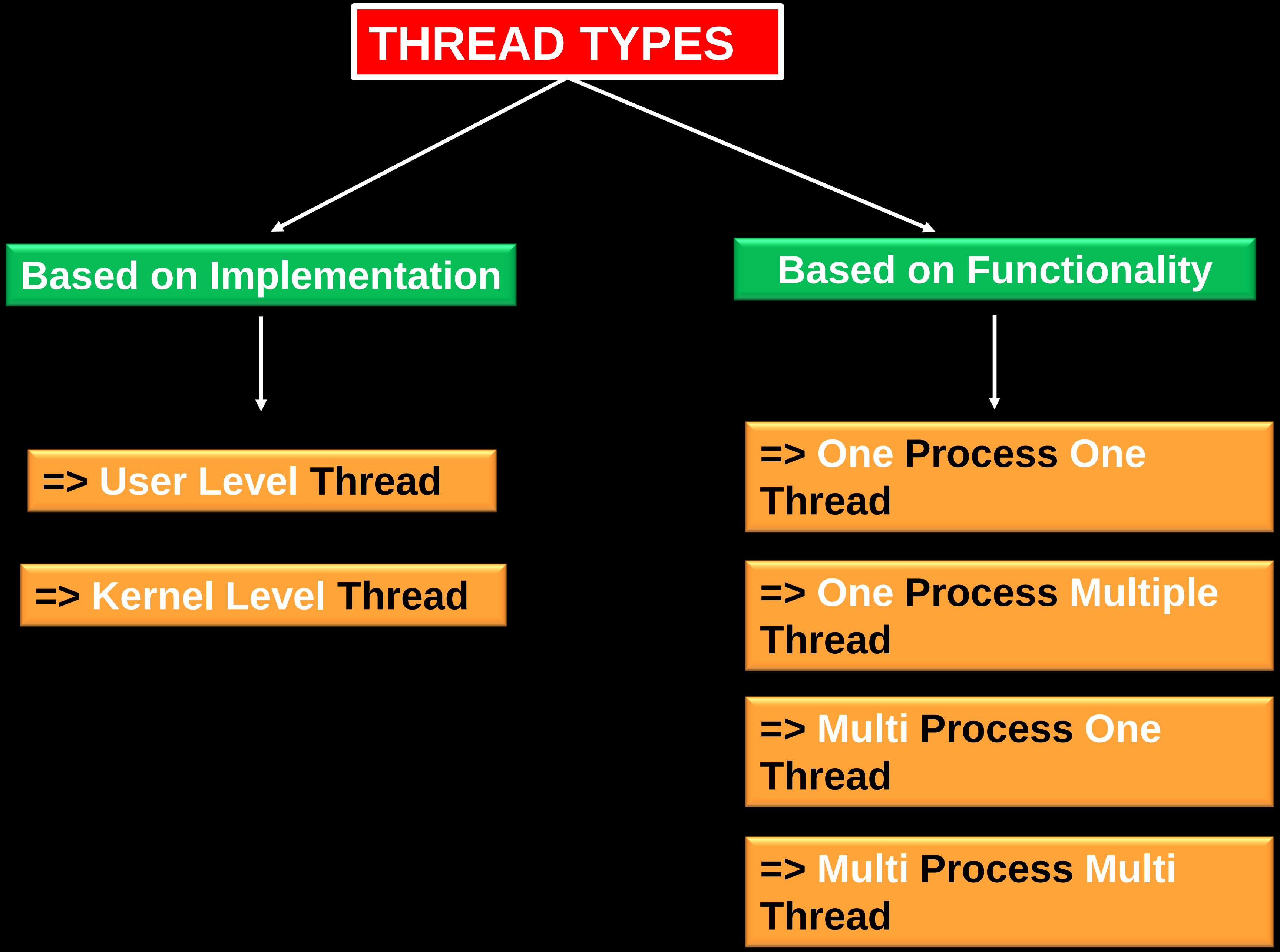
WAIT

SLEEP

DEAD

BLOCKED

Arivuselvan.K



THREAD TYPES

Based on Implementation

=> User Level Thread

=> Kernel Level Thread

Based on Functionality

=> One Process One Thread

=> One Process Multiple Thread

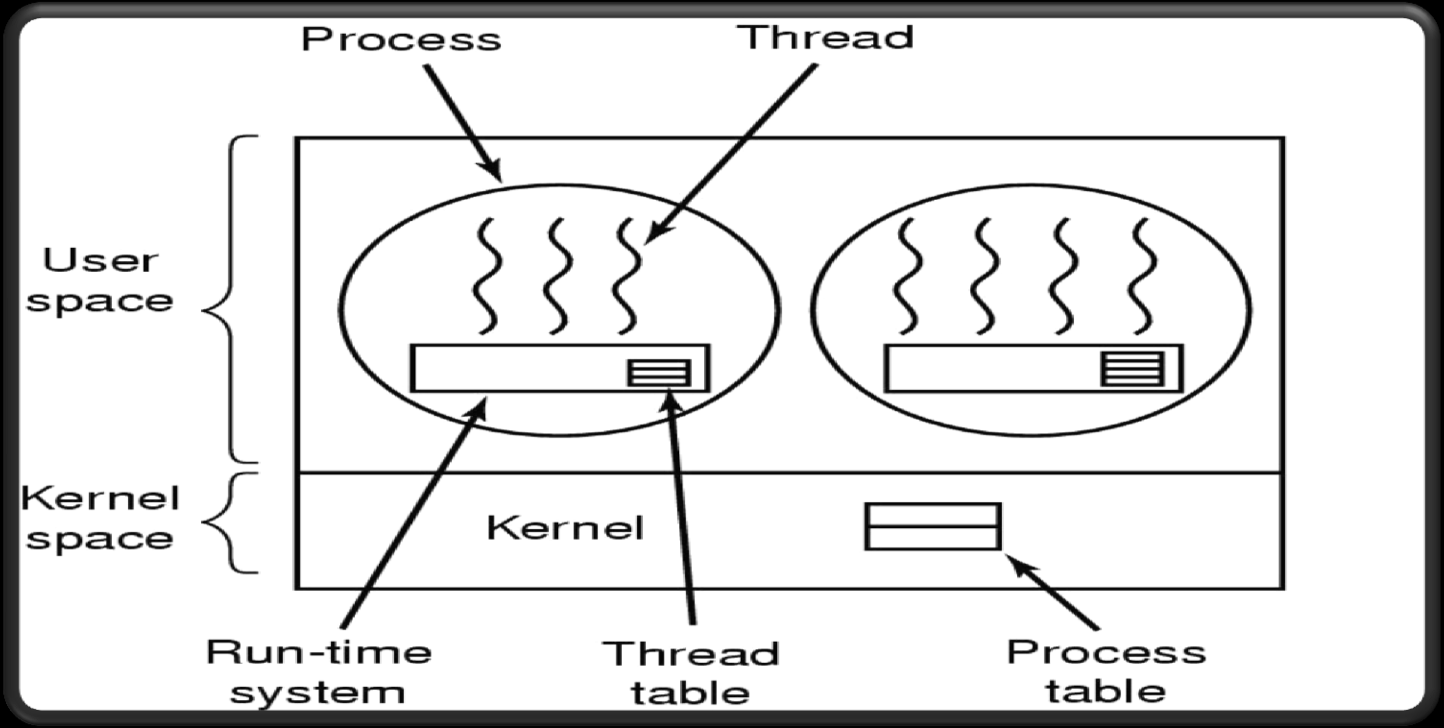
=> Multi Process One Thread

=> Multi Process Multi Thread

USER LEVEL THREAD

- Threads are loaded entirely in USER Space, Kernel is not aware about them

- Each process maintains its own private thread table which contains details about PC,STACK,REGISTER & STATE

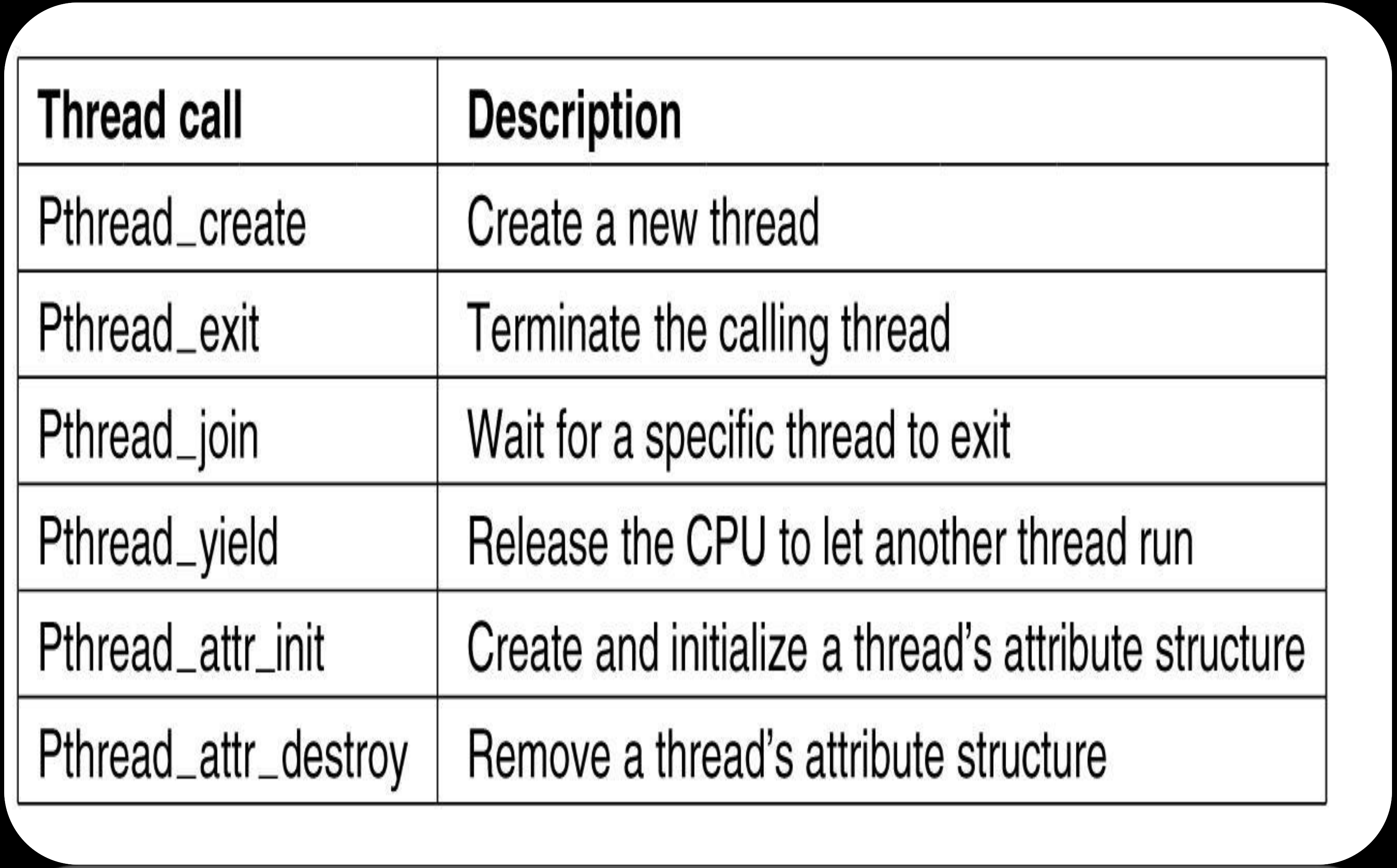


- Thread management done by user-level threads library

- Threads library contains code for:
  - Creating and destroying threads.
  - Passing messages and data between threads.
  - Scheduling thread execution.
  - Saving and restoring thread contexts.

- Three primary thread libraries:
  - POSIX Pthreads
  - Win32 threads
  - Java threads

Pthreads function calls



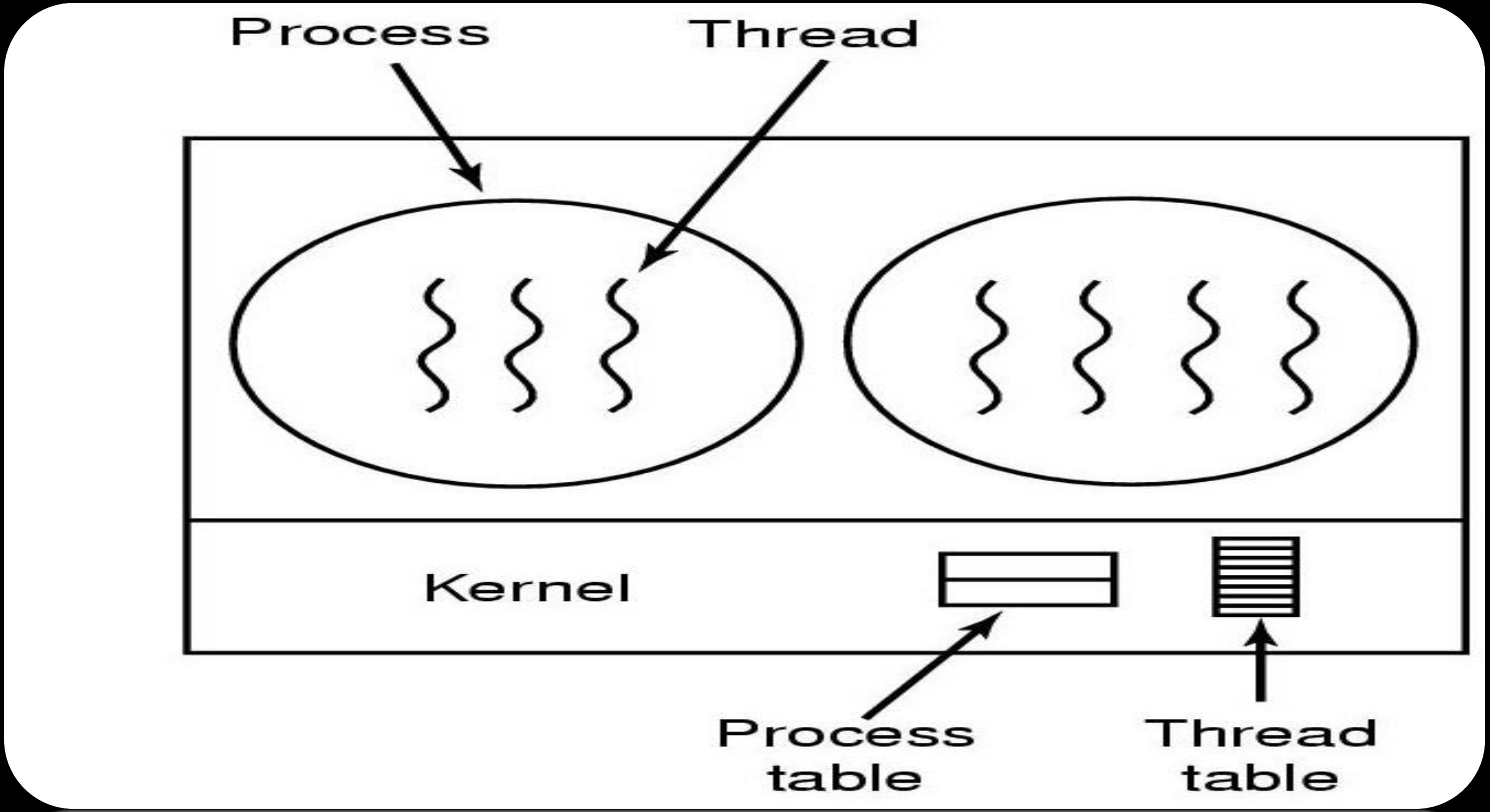
Thread call	Description
Pthread_create	Create a new thread
Pthread_exit	Terminate the calling thread
Pthread_join	Wait for a specific thread to exit
Pthread_yield	Release the CPU to let another thread run
Pthread_attr_init	Create and initialize a thread's attribute structure
Pthread_attr_destroy	Remove a thread's attribute structure

KERNEL LEVEL THREADS

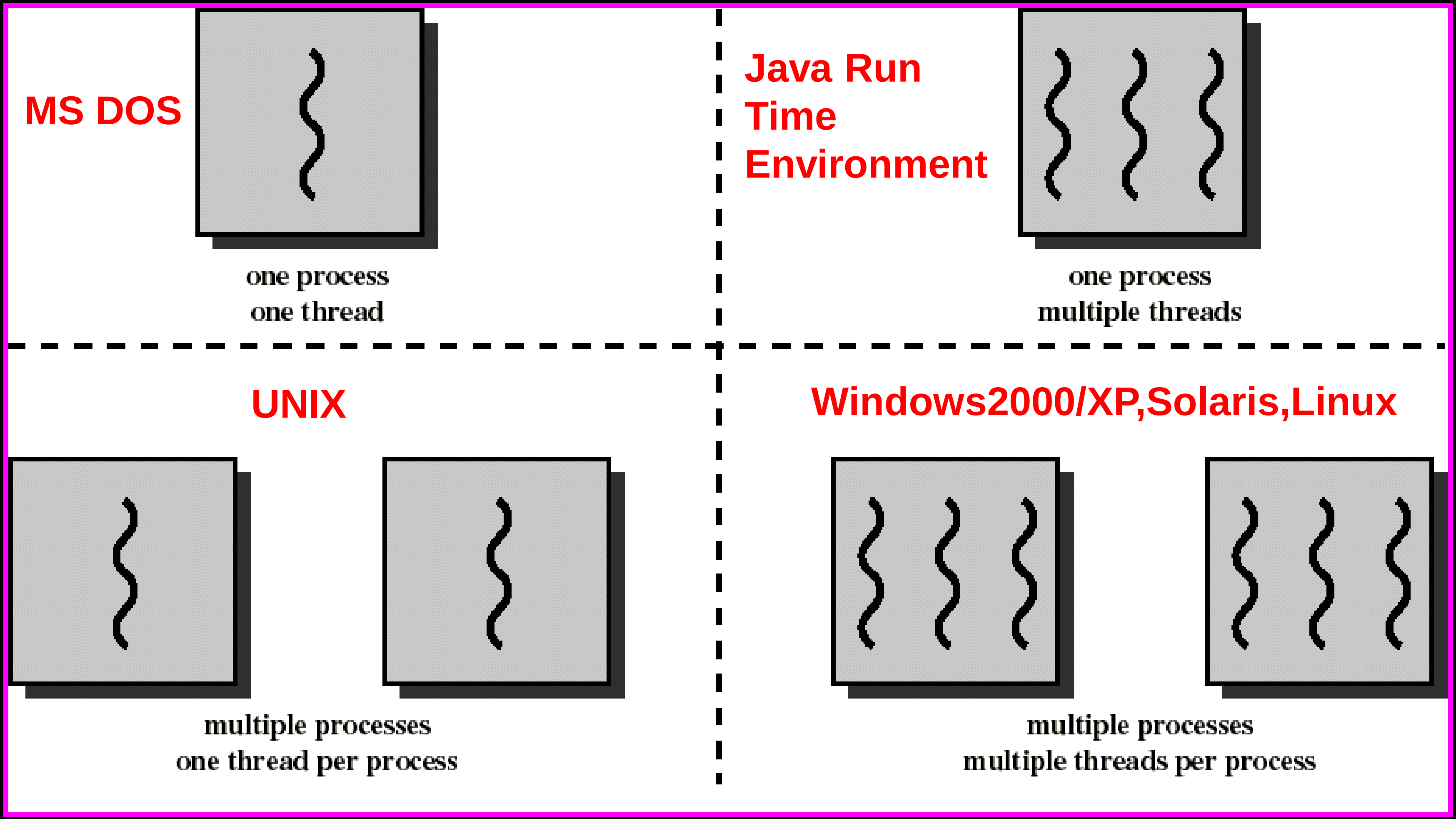
- The Kernel does total work of thread management

- No thread table in each process

- Kernel has thread table that keeps track of all threads in system



Combinations of Threads and Processes



MS DOS

one process
one thread

Java Run Time Environment

one process
multiple threads

UNIX

multiple processes
one thread per process

Windows2000/XP,Solaris,Linux

multiple processes
multiple threads per process

PROCESS VS THREADS

(1) Can't share the same memory area {address space}	Can share memory & Files
(2) Take more time to create a process	Takes less time to create
(3) More time to complete the execution & termination	Less time to terminate
(4) Takes more time to switch between two process	Takes less time to switch between two threads
(5) Communication between two process are difficult to implement	Easy to implement

PROCESS VS THREADS

(6) System Call required for communication	Not Required
(7) Process are loosely coupled	Tightly coupled
(8) Require more resource to execute	Fewer resource
(9) Not suitable for parallel activities	Suitable for parallel activities