(1) Difference between user-level vs kernal-level thre

(i) user-level threads are managed without kernal support by the run-time system

(i) kernal-level threads are supported and managed by operating system.

(ii) The scheduler cannot schedule the process properly as the kernal is unaware of user level threads

(ii) The scheduler handles the process better as the kernal is fully aware of kernal-level threads.

(iii) user-level threads are faster to create

(iii) Kernal-level threads are slower to create.

(iv) user level threads are more efficient

(iv) Kernal level threads are not so efficient

(v) The context switching time is less

(v) The context switching time is more

(vi) user level thread cannot take full advantage of multiprocessing.

(vi) Kernal-level threads take full advantage of multipro

(2) Difference between process and thread:

Process

(i) Process means any program in execution

(2) The process takes more times to terminate.

(3) It takes more time for creation.

Thread

(i) Thread means a segment of a process.

(ii) The thread takes less time to terminate

(iii) It takes less time for creation.

(4) It also takes more time for context switching	(4) It takes less time for context switching
(5) The process is isolated	(5) The Threads share memory
(6) The process does not share any data with each other.	(6) Threads share data with each other.

Q.3 Amdahl's law

→ Amdahl's law is a formula that shows the theoretical maximum speed up for a program when multiple processor or cores are added to a system. It was named after computer architect Gene Amdahl.

$$\text{Speed up} = 1 / [(1-P) + (P/n)]$$

where

P is the percentage of the program that can be parallelized

n is the number of processors or cores