Threads:

- The thread defines a sequential execution stream within a process (PC, SP, registers) The process defines the address space and general process attributes (everything but threads of execution)
- A thread is bound to a single process Processes, however, can have multiple threads Every process has at least one thread
- Changing running process is called a context switch
- Lighter-weight abstraction than processes.
- All threads in one process share memory, file descriptors, etc.
- Every thread operation must go through kernel - create, exit, join, synchronize, or switch for any reason
- User threads and Kernel threads are exactly the same. (You can see by looking in /proc/ and see that the kernel threads are there too.)
- They only need a stack and storage for registers therefore, threads are cheap to create.
- Threads use very little resources of an operating system in which they are working. That is, threads do not need new address space, global data, program code or operating system resources. (Advantage)
- Context switches between threads are faster than between processes. That is, it's quicker for the OS to stop one thread and start running another than do the same with two processes. (Advantages)
- Inter-thread communication (sharing data etc.) is significantly simpler to program than inter-process communication. (Advantages)
- Priority Q is good, but can lead to serious starvation, so not very good.
- Round Robin PQ is the best when it comes to scheduling.

References:

- https://mathlab.utsc.utoronto.ca/courses/cscc69s15/lectures/week1_2015.pdf
- http://www.scs.stanford.edu/15wi-cs140/notes/processes.pdf
- http://www.cs.iit.edu/~cs561/cs450/ChilkuriDineshThreads/dinesh's%20files/User%20and%20Kernel%20Level%20Threads.html