**Affinity Locking**

Setting Affinity

**COMSATS University Islamabad**

Sahiwal Campus



**Usama Sarwar**

FA17-BS(CS)-090-B

**Mr Umer**

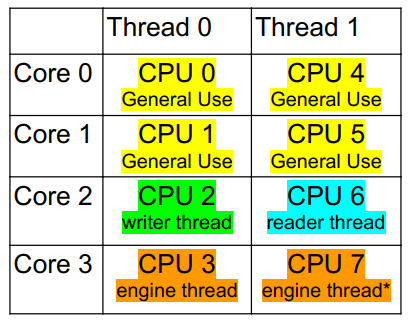
Operating System

December 03, 2019

**Allocating Threads and Cores**

Allocating threads and cores

Intel i7 with four cores and hyper-threading. "reader" and "writer" to be on the same core. "engine" to be on a core of its own.



# Output

Assigning cpu 7 to Thread[main,5,main]  
Assigning cpu 6 to Thread[reader,5,main]  
Assigning cpu 2 to Thread[writer,5,main]  
Releasing cpu 7 from Thread[main,5,main]  
Assigning core 3: cpus 3, 7 to Thread[engine,5,main]  
The assignment of CPUs is  
0: General use CPU  
1: General use CPU  
2: Thread[writer,5,main] alive=true  
3: Thread[engine,5,main] alive=true  
4: General use CPU  
5: General use CPU  
6: Thread[reader,5,main] alive=true  
7: Thread[engine,5,main] alive=true  
Releasing cpu 6 from Thread[reader,5,main]  
Releasing cpu 2 from Thread[writer,5,main]  
Releasing cpu 3 from Thread[engine,5,main]  
Releasing cpu 7 from Thread[engine,5,main]

# Code

public static void main(String... args) throws InterruptedException {  
**AffinityLock al = AffinityLock.acquireLock();**try {  
*// find a cpu on a different socket, otherwise a different core.***AffinityLock readerLock = al.acquireLock(DIFFERENT\_SOCKET, DIFFERENT\_CORE);**new Thread(new SleepRunnable(readerLock, false), "reader").start();  
*// find a cpu on the same core, or the same socket, or any free cpu.***AffinityLock writerLock = readerLock.acquireLock(SAME\_CORE, SAME\_SOCKET, ANY);**new Thread(new SleepRunnable(writerLock, false), "writer").start();  
Thread.sleep(200);  
} finally {  
**al.release();**}  
*// allocate a whole core to the engine so it doesn't have to compete for resources.***al = AffinityLock.acquireCore(false);**new Thread(new SleepRunnable(al, true), "engine").start();  
Thread.sleep(200);  
System.out.println("\nThe assignment of CPUs is\n" + **AffinityLock.dumpLocks()**);  
}