**Create 3 threads**

1st try:

Create a pthread with 3 threads with loops waiting with their individual semaphores.

2nd try:

Create 3 pthreads name A, B, and C with each blocked on its individual semaphores. Create 3 timers name A, B, and C each posts the associated semaphores to the threads.

3rd try:

Create 3 separate pthreads named A, B, and C. Each blocked on its individual semaphores. Create 3 timers name A, B, and C with timeout at 1, 2, and 3 seconds. Each timer posts the associated semaphores to the threads A, B, and C respectively.

4th try:

Create 3 separate pthreads named A, B, and C with 3 priority levels 1, 2, and 3. Each thread blocked on its individual semaphores. Create 3 timers name A, B, and C with timeout at 1, 2, and 5 seconds. Each timer posts the associated semaphores to the threads A, B, and C respectively.

**Create 2 threads**

Create 2 pthreads that count up a shared variable. Print the variable and the thread id each time. Access to the variable is protected with mutex. Run the example 20 counts.

create 2 pthreads call ping, pong with priority 1 and 2 respectively. Thread ping sleeps 1 second, thread pong sleeps 2 second. run each thread 5 loops. each time have thread prints its loop number.

**Create block diagram**

create a block diagram with 5 different data sets coming through the same I2C channel. The I2C use semaphore post to a distribution function. The distribution

* create a block diagram with 3 different threads that process 3 sets of data. Create 3 data I/O channels I2C, UART, SPI services. Each service use semaphore post to the threads.
* create the code for the example
* draw the block diagram such that the data should be coming into the services. Then the services provide the data to the thread.
* draw a diagram that 3 sets of different data come through the I2C. The I2C service post to a distribution pThread. The distribution thread then sort out the 3 different data packets and post separate semaphores to 3 processing threads.
* Draw the diagram such that there should be only 1 I2C service connects to the distribution thread. Theres should be 3 connection from distribution thread to 3 different processing threads.
* improve the diagram so there are 3 separate lines from the distribution thread going to 3 separate processing threads.
* there should be only 1 I2C service with only 1 line from it to the distribution thread
* up date the diagram so that all 3 data set come into the I2C service. 1 line from I2C service to distribution. 3 lines from distributions to 3 separate processing threads.
* improve the diagram to indicate a 3 data set go into the I2C service with indication that each data set goes to different processing thread. Each processing thread will receive only 1 data set which was separated by the distribution thread.
* generate pThread code with the use of semaphore for this latest diagram.