1) Procedure for creating a µC/OS II project:

Our task is to create a uC/OS II project on KEIL IDE for MicroLPC2148 board from MicroEmbedded Technologies.

This procedure is designed for Keil uVision V4.20. Make sure you have a similar or higher version of Keil installed on your machine and locate the "**Raw uCOS-II**" folder. Also note that you **must create a new folder** for every new project.

Follow these steps by the end of which you will have executed a rtos project.

1. Open the '*Raw uCOS-II*' folder. Copy the folder '*app*', '*uCOS-II*', '*include*' and '*ScatterFile.sct*'.
2. Create and name a folder at any location you wish to. E.g. Let's say we name that folder as 'demo1'.
3. Open the demo1 folder and paste the contents you copied from step 1.
4. Now minimize all the windows and open *Keil uV4*.
5. Go to '***Project -> New uVision Project***'.
6. You will be asked to select a location for the project and name of the project. Browse and open the demo1 folder. Then give a name to the project (say "demo1") and click save. The name of the project and the folder need **not** be the same. The name of the project "demo1" has **no** extension, type only the name.
7. Next, select the device '***NXP (founded by Philips) -> LPC2148***'.
8. You will be asked if you wish to copy the **startup** code to the project. Click **NO.**
9. You should now have an empty project on your screen. In the top left corner of the screen click on the + symbol next to '***Target1***'. You should see '***Target1 -> Source Group1***'.
10. Right click on '***Source Group 1***' and click on '***Remove group Source Group 1 and its files***'. So now you are left with only '*Target1*'.
11. Now, right click on '***Target 1***' and select '***Add Group***'. Rename that group to '***uCOS II***'.
12. Similarly, add two more groups called '*app*' and '*includes*'. This makes your project structure look something like this

    Target 1 + → **uCOS II**
    → **app**
    → **includes**

13. Now right click on the group '***uCOS II***' and select "***Add files to group uCOS II***'. Browse into the uCOS II folder. Change the '***Files of type***' option to '***All Files*.***'. Select all the files or press CNTL+A. Then click on ADD and close.
14. Similarly, right click on groups '*app*' and '*includes*' and add all the files of those respective folders. So now we have all the files from the folders from step 3 added to the respective groups in Keil.
15. Now, go to '***Project -> Options for Target 1***' or right click on '***Target 1' and then click on 'Options for Target 1***'.

16. The '*Options for Target 1*' window will be open. Go to the **'*Output*'** tab and tick on the **'*Create HEX file*'** option.
17. Then go to **'*Linker*'** tab and make sure that the **'*Use memory layout from target dialogue*'** is **un-tick**. Click the browse or the '...' button in front of the Scatter File option and browse and add the **"*ScatterFile.sct*"** from step3.
18. Then go to **'*C/C++*'** tab. In the **'*Include Paths*'** option click browse or the **'...'** button. Click on **'*New (Insert)*'** and click on the browse or the '...' button. Browse and select the include folder in *".....\demo1\include*".
19. Our uCOS II project is now completely set-up. To verify everything is **OK** go to **'*Project -> Build Target*'**. After compilation you should see that the HEX file is been created and there are *0 errors*. Ignore the warnings. You can now proceed with the programming.
20. Expand the '*Target1 -> app*' group and open the file '*app.c*'. this is th file you need to modify for your specific application.
21. In this file you file find that the provision for 4 tasks (1 parent and 3 child tasks) has already been made for you, along with their priorities, stacks and task definitions along with board specific initializations.
22. Find the section where AppTask0 is defined (most probably line 147). Inside the *while(1)* loop type your code to toggle *LED0* and call *OSDelay( )* function.
23. Similarly in the AppTask1 and AppTask2 section type your code to toggle LED1 and LED2 respectively.
24. Compile the code and generate the HEX file.
25. Burn the *demo1.hex* file into the microcontroller using the Flash Magic utility.

Happy programming ;)

2) Verify the results.

1. **MailBox.**

   - In this program two tasks (**AppTask0 and AppTask1**) are created.
   - The mailbox (**TxMbox**) is also created .
   - When the user runs the program, Task 0 waits for the user to enter a key from the **PC keyboard** (through serial terminal of flash magic).
   - This **key** (ascii number) is accepted by the task0 and send to task1 using a mailbox.
   - The Task1 receives this key from the mail box and displays it on the LCD. (observe the character displayed on LCD)

2. **Message Queue.**
   - In this program two tasks (**AppTask0 and AppTask1**) are created.
   - A Message Queue (**MsgQ**) is also created. An array of 5 message is already in the program.
   - When the user runs the program, Task 0 posts the messages in the message queue with a delay of 2 seconds. All these message are now queued.
   - Task 1 received the message from the queue, after delay of 3s and 800ms is present.
   - The user can observe the sending and receiving of the messages. Also user can change the timing parameters to observe scheduling of the tasks and reception of messages.
   - The user can observe the program on the serial terminal of flash magic software at 9600 baud.

3. **Multitasking**
   a) **Multitasking 1**
      - In this programs 4 tasks (AppTask0 to AppTask3) are created.
      - In every task a LED is blinked at different time intervals.
      - The user can change the time of delay and observe the scheduling of the tasks and blinking of the LEDs.

   b) **Multitasking 2**
      - In this programs 3 tasks (AppTask0 to AppTask2) are created.
      - In Task1 a LED is blinked at a interval.
      - In Task 2 a character is displayed on the LCD.
      - In Task 3 a stepper motor I rotated.
      - The user can observe the timing of the blinking of the LED, the time of character display on LCD and rotation of the stepper motor.

- The user has to observe the simultaneous working of the 3 tasks and also the scheduling.

4. **Mutex**
   - In this program 3 tasks (AppTask0 to AppTask2) are created.
   - A Mutex (LCD_Mutex) for LCD(resource) is also created.
   - The three Tasks fight for the LCD resource. Depending on the scheduling and the time delays, the mutex is acquired by individual tasks.
   - The task that has the Mutex will have the resource.
   - After the task is over the mutex is released and so is the resource.
   - The user can observe this behaviour by the data that is printed on the LCD.
   - The user can try and make changes to alter the behaviour.
   - The user can observe the program on the serial terminal of flash magic software at 9600 baud.
   -

5. **Semaphore Signalling**
   - In this program 2 tasks are created(AppTask0 to AppTask1).
   - A Semaphore (Sem) is created for synchronizing the tasks.
   - Task 0 is waiting for the user to press the key on the matrix keypad on the board. It is holding the semaphore .
   - When the user will press the key the semaphore will be released (posted) and will be acquired by Task1.
   - When Task1 will get the semaphore it will perform the operations until then it will be blocked.
   - The user can observe the program on the serial terminal of flash magic software at 9600 baud.

6. **Semaphore Syncing.**
   - In this program 2 tasks are created(AppTask0 to AppTask1).
   - A Semaphore (Sem) is created for synchronizing the tasks.
   - The 2 Tasks will be holding the semaphore, simultaneously.
   - Task 0 will wait for the user to press the key on the on board keypad. Once the user presses the key, the semaphore will be posted (or signalled).
   - Task 1 will then be pending for the semaphore till then. Once it gets the semaphore it will then wait for the user to press the key.
   - Once the user presses the key the semaphore will be signalled for task 0 to performs its operations.
   - The user can observe the program on the serial terminal of flash magic software at 9600 baud.

- The user can observe the synchronization by using the keypad and observing the scheduling.