For implementation of kernel module, when userapp starts running, it registers as a task in the kernel space. The core of implementation is dispatcher thread and yield function. For yield function, it is necessary to figure out the first time yield call or later yield calls for each task. In this way, next_task (unsigned int64) is used as reference to determine whether it is the first time to call yield or not. In the Yield, the timer for next interrupt has to been set according to current time. Another core is dispatcher thread, it is necessary to pick the highest priority tasks among all potential valid tasks. It is also mentioned that in deregister function, it is also necessary to wake up dispatcher otherwise the system would freeze.

For implementation of userapp.c, the basic frame is the same as that in ppt. Basically, there are three parameters for each task, the number of periods, the length of period and computation time of each period.
And It is required to input two arguments, the first, number of periods and another parameter which determined the effective computation time and length of period. The details can be checked in comments

Ex: ./userapp 3 1 which means the task would run 3 periods and period and computation time is determined by 1.

After make and insmod mp2.ko. **Just execute** *test.sh* which contains a test example of it From the example, it can be observed that two tasks take in turns based on the length of the period. The yield function and timer interrupt work correctly as well. The dispatcher chooses the highest priority task to run. Finally, both tasks deregister successfully.