2. Dynamic Instrumentation in Kernel Modules

key words: Linux kernel RB tree, Linux module and device driver, Kprobe, x86's TSC (Time stamp counter) to measure elapse time, Multi-threaded programs.

- Develop a kernel module named as "RBprobe", that uses kprobe API to add and remove dynamic probes in any kernel programs.
- With the module's device file interface, a user program can place a kprobe on a specific line of kernel code, access kernel information and variables.
- Test program reads in kprobe request information from console and then invokes RBprobe device file interface to register/unregister a kprobe at a given location of rbt_drv module.
- When the kprobe is hit, the handler should retrieve few trace data items in a buffer such that they can be read out via RBprobe module interface.
- The user input request consists of the location (offset) of a source line of code on the execution path of read and write functions of rbt_drv.
- The buffer is with a fixed size and holds one set of trace data, i.e. any old trace data will be overwritten when new trace data is generated.
- The trace data items to be collected by kprobe handler include: the address of the kprobe, the pid of the running process that hits the probe, time stamp (x86 TSC), and all rb_object objects traversed in the RB tree while performing the corresponding functions.
- The read and write operations of RBprobe device can be defined as:

  write: to register or unregister a kprobe. The location (offset) of the kprobe is passed in the buffer **buf* along with an integer flag. A kprobe is registered if the flag is 1, or unregistered if 0.

  read: to retrieve the trace data items collected in a probe hit and saved in the buffer. If the buffer is empty, -1 is returned and errno is set to EINVAL.
- Using proper synchronizations, you can control how the 4 threads invoke the operations to rbt device file which can result in a hit at the kprobe point.