Moumita Laskar
ASU ID: 1204363181

Assignment 1

RB Tree and Dynamic Probe in Linux Kernel

Introduction

Implement a Kernel Module of RB Tree and have Dynamic Probe (kprobe) on the operations of RB Tree.

Part 1 of the assignment is based on inserting and deleting values on RBTree module rbt530_dev". The balanced BST is based on key value and data is user specific.
Insert:- Whenever a write happens, a new node is created and inserted on the RB Tree, provided it has a unique key. If the key already exists on the tree, the node is replaced with a new node. If data is 0, the node with key value provided is being deleted.
Read:-The RB Tree can be read in ascending or descending order based on the user's choice via ioctl call. If the argument of the ioctl is "ascending", the RB Tree content is displayed in ascending order, if "descending", its displayed in descending order. For any other values the ioctl implementation will defaults to "descending"

Part 2 of the assignment is the ability to put dynamic probe on the RB Tree module. A module "RBprobe" is created to accomplish this task. Users can dynamically give the offset or symbol name to probe the RB Tree from the user interface along with 1 or 0 to register or deregister the address to be probed. The read function of the module reads the RB Tree content and displays it into the user console.
To test the functions of RB Tree and Probe, two instances of RB Tree module is created , with two different minor numbers for the same major number of "rbt530_dev", namely "rbt530_dev1" and "rbt530_dev2".Each instance of rb tree is operated by two threads.Apart from these four threads, fifth thread is created for the functionality of probe i,e take the address and flag to register/deregister and display the content of the tree in user space.

Mandatory Requirement:

- RBT_530 module can be independently compiled and run with the Makefile in Part1 folder along with rbt_tester_probe application.
- RBProbe module requires RBT_530 module and rbt_tester_probe application to probe the RBTree instructions. However the Makefile in Part2 is meant to compile RTprobe.c independently.

Content of the Assignments

1. C files

/*User space test program for two modules. Executable is named rbt_tester_probe */
Main.c
/*Driver module program for Dynamic Probe */
RBprobe.c
/*Driver module program for RBTree 530 */
RBT_530.c

2. Header files

/*ioctl declaration for setting up ascending and descending order for RBTree*/
RBT_530.h
/*shares the definition of Tree Nodes structure between tree and probe modules*/
RBT_Nodes.h

3. Shell scripts

/*makes modules and tester app and copies the files to /tftpboot/ if they are compiled from same directory.
Buildscript.sh
/*download *.ko from /tftpboot/ and install the modules using insmod */
Install.sh
/*uninstall the modules from the kernel using rmmod command */
Uninstall.sh
/*uninstalls, installs and starts the main program */

4. /*Makefile compiles the C files builds the modules for Galileo gen 2 board */
Makefile. The Makefile in Part1 can include RBprobe with small edit to build from same location.

Steps

Compilation and Installation Steps

/* On Host machine*/ <buildscript.sh>

1. make clean
2. make (to compile both kernel module and tester app)
3. cp *.ko /tftpboot/
4. cp rbt_tester_probe /tftpboot

/*Target Machine*/ <./uninstall.sh> followed by <install.sh>
<edit install.sh to added ascending or descending while executing ./rbt_tester_probe>

1. cp /tftpboot/*.ko to desired location
2. Lsmod |grep RBT_530
   if (installed)
   Rmmod RBT_530
   Insmod RBProbe

3. Lsmod |grep RBprobe
   if (installed)
   Rmmod RBprobe
   Insmod RBProbe
4. ./rbt_tester_probe <ascending|descending>

Helper scripts

/*Host Machine

1. Execute ./buildscript.sh in host machine

/* Target (Galileo gen 2)

2. ./install.sh , downloads the files from /tftpboot/ and installs the modules and executes tester program
3. ./uninstall.sh , to uninstall the kernel modules

Input

1. RBTree The key and data are randomly generated in user space and are written to buffer which is then received in kernel module RTB530.
   Users have the choice to specify address to probe along with option “ascending” or “descending” to display the tree information in ascending or descending order.

2. KBprobe: KbProbe thread is created and waits for user input to register/deregister probe on RBTree specified by address and a flag. Flag specifies to register or deregister the probe for particular address. Currently the first two probes are hardcoded to insert and read the function of RBTree.
   Users can type either add <address> <flag> to register/deregister or stop to stop the program.

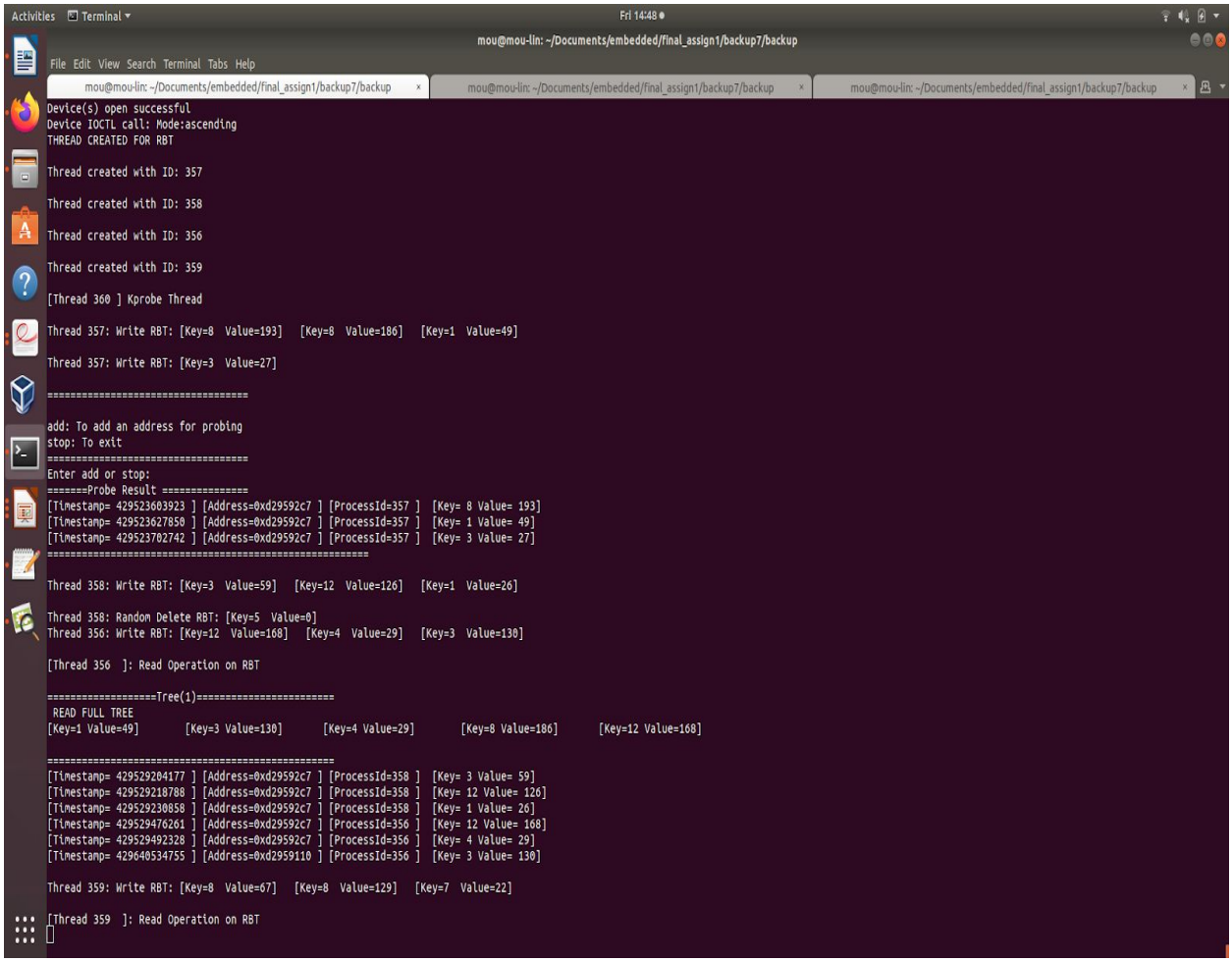
Processing

1. RBTree: The threads randomly performs insert/read tree and display(dump entire tree) based on flag for ascending or descending.

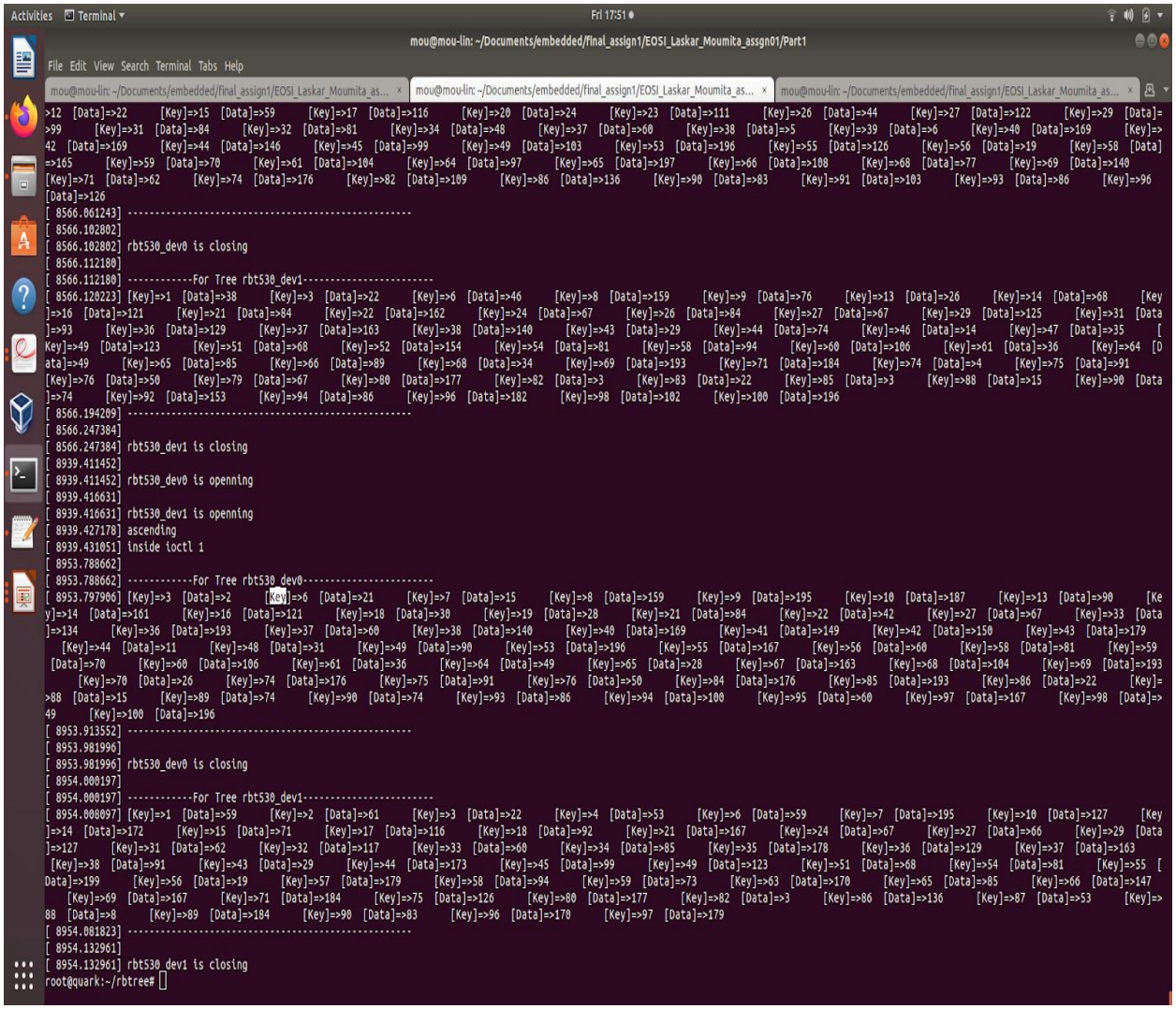
2. KBProbe: User thread, wait for user to enter address and flag and simultaneously kprobe module performs probe on insert and read from addresses which are currently hardcoded.

Output

1. RBTree -
   - Operations read/insert/display of RBTree is displayed on the console.
2. From Kprobe
   - Display the information of timestamp when the probe was hit, PID of the thread which executes the command and key and data for which the probe was hit.
3. Dmesg gives the output of the RBTrees module. It shows the content of the RBTrees when the program closes



Output with RBprobe



Output of RBT_530 module(dmesg output)

Deliverables:

Folder EOSI_LASTNAME_FIRSTNAME_assgn01

Readme

Part1

- Main.c
- Makefile
- RBT_530.c
- RBT_530.h
- RBT_Nodes.h

- This Makefile creates module RBT_530.ko for read/write and ioctl operations

Part2

- RBprobe.c
- Buildscript.sh
- Install.sh
- Uninstall.sh
- Makefile
- This Makefile will is required to have RBT_530.c in the same folder as there is symbol dependencies.