Virginia Tech ❖ Bradley Department of Electrical and Computer Engineering

ECE 4984 / 5984 Linux Kernel Programming
Spring 2017

Small Project: Kernel Modules and Data Structures

1 Introduction

The goal of this project is to develop your first kernel module, and to study the manipulation of the following frequently-used kernel data structures: linked lists, queues, maps, and red-black trees.

The following concepts from the course will be put in practice in that project:

- Kernel module development;
- Kernel data structures.

2 Requirements

The requirements for this project are to write a *single* Linux kernel module manipulating the above-mentioned kernel data structures. The module takes one integer parameter `dstruct_size`. The module should compile against a v4.0 kernel. The virtual machines used in project 1 can be re-used for that project, it is available here: `http://bit.ly/2bKfKVO`.

The module should include functions manipulating the following data structures:

Linked lists : (A) constructs a linked list containing `dstruct_size` random integers; (B) prints on the kernel log the content of the list through the use of list navigation functions; and (C) destructs the list and free its content. It is asked to write two versions of the linked list exercise: one executing (A), (B) and (C) in the same function, the second calling (A), (B) and (C) each in a different function.

Queues : creates a queue and enqueue `dstruct_size` random integers, then dequeue these elements, print them, then free them if needed.

Maps : creates a map and insert in it `dstruct_size` random integers. The `id` value associated with each element should be constrained into a specific range defined by a low and high bound, clearly indicated in the code. It is asked to print the content of the map and the id associated with each element before freeing the data structure.

Red-black trees : creates a `rbtree` with nodes being random integers, indexed by the integers values. It is asked to insert into the tree `dstruct_size` random integers (while being careful not to insert duplicates). Next, print the tree on the kernel log. Then, a loop should iterate over an random integer range that is a valid subset within the current tree; for each of value found in this range, indicate (by printing to the kernel log) if it is present in the tree and remove the element. Next, print the tree again. Finally all remaining elements should be removed from the tree and the data structure freed if needed. Note that with the red-black tree implementation of Linux, it is needed to implement the search and insert functions.

Note that since Robert Love's book publication, the interface for the map data structure has changed (in 2013). Information about the new interface can be found here:
`https://lwn.net/Articles/536293/`

Moreover, as the textbook does not give extensive information about the use of red-black trees, note that additional help can be found in the kernel documentation itself:
`https://www.kernel.org/doc/Documentation/rbtree.txt`.

Finally, one way to generate random numbers in the kernel is through the use of the `get_random_bytes()` function:
`http://lxr.free-electrons.com/source/drivers/char/random.c?v=4.0#L1233`

3 Results to be handed

The deadline is as follows:

- **For the graduate students:** 02/09 11:59 PM;
- **For the undergraduate students:** 02/16 11:59 PM;

The following is expected to be handed by the deadline: sources for the kernel module plus associated `Makefile`.

All of this should be contained in a *tarball*, with the following format: `<VT ID>.project2.tar.gz`. For example: `johndoe.project2.tar.gz`