Spring 2020

CS 680-002: Linux Kernel Programming (Revised for Remote Learning)

Andrew Sohn

CS680 Linux Kernel Programming - Syllabus for Spring 2020

Andrew Sohn, Computer Science Department

Tues, 4/21/2020

List of changes due to COVID-19 and online Webex-based lectures since the announcement of class cancellation on 3/12/2020:

- Reduced the number of homework to **8** from 10.
- Reduced the last two weeks of lectures on networking to half so that the time can be used for earlier materials that were not properly discussed at length in detail due to the sudden change of instruction.
- Sunday class was offered once (4/19/2020). Half the number of students were unable to log on to Webex due to heavy traffic on webex during the entire class period of 4-5:30 pm, Thur, 4/16/2020. I was able to log on to webex and stayed on the entire class time (an hour and 30 minutes) to check how many students were able to log on. After two straw polls, I emailed them to come on Sunday, 2 pm, 4/19/2020 for class.

- Class Web page: http://web.njit.edu/~sohna/cs680
- Homework submission page: http://canvas.njit.edu
- Instructor: Andrew Sohn, GITC 4209, (973)596-2315, email: sohna at njit dot edu
- Office Hours: Tue 10:30-11:30 am, Thur 2:30-3:30 pm
- TA/Grader: Yongjian Wang yw662 _at_ njit _dot_ edu, Office hours and location: TBA
- Class time and location: See the registrar's page https://uisnetpr01.njit.edu/courseschedule
- Kernel version 5.4 as of 1/21/2020: download the latest version at kernel.org
- Book required: W. Mauerer, Professional Linux Kernel Architecture, 2008, Wiley, ISBN:9780470343432 (10 years old but it's still "the latest" or "the most recent") - the NJIT bookstore has no hardcopy as it's too old. Get it elsewhere.
- Books recommended:
  - Understanding the Linux Kernel, Third Edition, 2006, Bovet and Cesati, O'Reilly, ISBN: 0-596-00565-2 (very old but you still learn a lot).
  - Intel 64 and IA-32 Architectures Software Developer's Manual (5038 pages!) as of 1/21/2020. https://software.intel.com/sites/default/files/managed/39/c5/325462-sdm-vol-1-2abcd-3abcd.pdf
  - Computer Systems: A Programmer's Perspective, 3/E (CS:APP3e), Randal E. Bryant and David R. O'Hallaron, Pearson (July 6, 2015), ISBN-13: 978-0134123837, ISBN-10: 0134123832.
- Grading: **8** kernel programming assignments (20%), Test 1 (20%), Test 2 (20%), final exam (40%). Homework must be done individually, will not be accepted after the due date. Submit it on time. Do not email your homework. Emailed homework will be discarded. Do not ask for exceptions. No exceptions will be made. If you are unable to do the first three homework which are the easiest, this class is not for you.
- Setting up lxr - linux cross referencer - for source code traversing. See http://lxr.sourceforge.net/en/index.php for setup instructions. If you are unable to set up lxr

on your box in the first two weeks, you have to seriously think if this course is for you as you need lxr to study the source code and do homework.

- NJIT policy on missed exams: There will be no make up exam(s). You must plan your semester accordingly. If you should miss the exam(s) due to emergency, go to the Dean of students and explain your situation as to why you had to miss. Dean's memo will be necessary but not sufficient to consider for handling your missed exam(s). This is the NJIT policy for missed exams. No other policy will be applied.
- You are required to read the NJIT Policy on Academic Integrity for this course: https://www.njit.edu/policies/sites/policies/files/academic-integrity-code.pdf
- Invitation to your social network: Do not ask to be connected to your social network. My policy is not to join anyone or any social network for any reason. Once it's out, it will be out there forever. You will never be able to take it back. Do at your own risk.

Lecture Schedule by Week (may change based on class pace)

1. Introduction: Lecture Note 1
    - LAMP, virtualization, containers, datacenter computing infrastructure,
    - Review of Intel architecture based on Intel 64 and IA-32 Architectures Software Developer's Manual (4684 pages! as of 1/16/2018),
    - Short recap of Intel and AT&T Linux assembly
    - "Hello, World!" in Intel assembly, Linux assembly, assembly in-line programming with C
    - Setting up LXR (Linux cross referencer http://lxr.sourceforge.net/en/index.php) on your laptop
2. Preparatory steps: Lecture Note 2
    - Compiling the kernel, which will take hours the first time
    - Module programming - see The Linux Kernel Module Programming Guide
    - Writing your own system calls, adding to the syscall_64.tbl
    - Booting - machine BIOS, disk MBR, Grub Linux loader, preliminary setup (setup(), startup_32/64() 1 and 2)
    - Overview of kernel startup and initialization - start_kernel()
3. Memory - initialization
    - Overview of memory spaces: logical segmentation, linear virtual, actual physical
    - Detecting BIOS-provided e820 physical RAM map: detect_memory()
    - Converting to memblocks: setup_arch(): e820__memory_setup(), e820__end_of_ram_pfn(), e820__memblock_setup(), init_mem_mapping(), initmem_init()
4. Memory - paging, buddy system, setting up page directories (global, upper, middle), tables and PTEs
    - (N)UMA, nodes, zones (DMA .. Normal), memory types (Unmovable .. CMA .. Isolate), free_areas
    - Setting up buddy system: x86_init.paging.pagetable_init(), paging_init(), zone_sizes_init();
    - Allocating 1 to 1K contiguous pages from buddy system: __get_free_pages() to __rmqueue_smallest()
    - Freeing pages: free_pages() to __free_one_page()

- Setting up slabs for small memory objects of 8 bytes to 8 KB: mm_init() to kmem_cache_init() for initializing general purpose kmalloc_caches():

5. Memory - setting up kmalloc_caches (slabs) for small objects of 8 bytes to 8 KB to large non-contiguous memory space
    - Setting up slabs for small memory objects: mm_init() to kmem_cache_init() for initializing general purpose kmalloc_caches():
    - Allocating small memory chunks: kmalloc() of 8 bytes to 8 KB from slabs
    - Freeing small memory chunks: kfree() to return to slabs
    - Large non-contiguous memory, process address space: vmalloc() and vfree() if time permits
    - User malloc(): will discuss briefly or provide pointers to read on your own as it's simple compared to __get_free_pages and kmalloc if time permits.

Test 1, Tue, 2/18/2019, for an hour and 15 mins, the same classroom

6. Process - structrures, organization, initialization
    - Structures: thread union, thread info, stack, task, and thread struct, PID0 (swapper)
    - Macros to initialize PID0: INIT_THREAD_INFO(init_task), INIT_TASK(), INIT_TASK_TI(), ...
    - Initial hardcoded structs: init_task, init_stack, init_mm, init_fs, etc.

   Process - creating the first five kernel threads
    - Creating kernel threads: kernel_thread() - copy and insert insert into RB tree, create_kthread(), hot plug thread
    - P0 creating PID1 (init) and PID2 (kthreadd) using kernel_thread(),
    - P1 and P2 creating PID3 (softirqd), PID4 (migrationd) using create_kthread() through kthread_create_list
    - Hot plug threads for creating P3 and on using kthread
    - Sync mechanisms between P0, P1, and P2 for creating P1 and P2; and between (P1,P2) and (P3,P4,...) for creating P3 and on.
7. Process - process scheduling (do_fork() to schedule() to rb_entry())
    - update_curr(): priority, nice value, weight, delta, weighted delta, actual runtime, virtual runtime
    - schedule() - configurable scheduling policies, fair_sched_class, computing vruntime based on priority (nice values) and weighted delta.
    - Scheduling processes with red-black tree: pick_next_task(), put_prev_task;

   Process - process switching (schedule() to __switch_to())
    - Context switches
    - Switching to suspended process
8. Process scheduling and switching continue

   Interrupts - PICs, APICs, exceptions (traps) and hard interrupts, IDTs
    - Hardware organization: Programmable Inerrupt Controller, interrupt vectors, CPU interrupts and interrupt acknowlege to interrupt handlers.
    - Initialization: sort_main_extable(), trap_init(), init_IRQ(), softirq_init(), initializing IDTs, irq_desc and softriq_vec
    - Exceptions: exception handlers, do_trap() to do_exit() if no trap handler
9. Interrupts - timer interrupts, soft interrupts, ksoftirqd, run_timer_softriq

- Hard interrupts: registering interrupt handlers request_irq() to do_IRQ() to handle_level_irq(), edge vs level trigger
- Hard interrupts: IDT handler common_interrupt, interrupt_entry, do_IRQ(), device driver action->handler(), raising softirq invoke_softirq(), softirq_vec action
- Soft interrupts: hardirq (schedule,producer,top half) vs softirq (action,consumer,bottom half)

10. Interrupts contiune - PICs, APICs, exceptions (traps) and hard interrupts, IDTs
    - ksoftirqd (kernel thread 3, P3), do_softirq(), softirq action, run_timer_softirq() as an example

Test 2, Tue, 3/31/2019, for an hour and 30 mins, Canvas

11. File system - virtual file system, block IO, elevator scheduler, device driver, softirq, timer, delayed work, kblockd_workqueue
    - initialization: vfs_caches_init, mnt_init: Virtual file system VFS
    - Registering, mounting
    - Scheduler - completely fair queuing
12. File system - device driver, Ext4 example (vfs_read() to to scsi_dispatch_cmd())
    - submit_bio(), single queue HDD vs multi queue SSD
    - The big loop in time and space: timer, delayed work, kblockd_workqueue
13. File system
    - Ext4 disk organization: MBR, superblock, group descriptors, bitmaps, inode table, data blocks.

    Networking - receiving packets
    - receiving packets: softnet data, input_pkt_queue, process_queue, budget
    - receiving packets: NIC, ISR, Softirq, IP, TCP, Inet, BSD, User
    - User BSD sockets read(), tcp_recvmsg, , Inet socks, TCP and IP layer
    - Softirq: net_rx_action, ip_rcvmsg, tcp_recv_msg
14. Networking - sending packets
    - sending packets: output_queue, User, BSD, Inet, TCP, IP, Softirq, Qdisc, ISR, NIC
    - User BSD sockets, Inet socks, TCP and IP layer
    - Qdisc - p/bfifo packet/byte based FIFO queueing discipline, ISR interrupt service routine, NIC
    - Softirq - dequeue, transmit, ISR interrupt service routine, NIC, requeue, delayed timer
15. Final exam (week15): See the registrar's page: http://www.njit.edu/registrar

Read the following carefully to make an informed decision on whether this course is for you: