Virginia Tech ❖ Bradley Department of Electrical and Computer Engineering

ECE 4984 Linux Kernel Programming
ECE 5984 Advanced Linux Kernel Programming
Spring 2017, Course Syllabus

---

1 Course Reference Numbers (CRNs)

- **Physical presence on VT Blacksburg campus:**
  - ECE 4984 - Linux Kernel Programming: **19480**
  - ECE 5984 - Advanced Linux Kernel Programming: **19494**
- **Off-campus online through WebEx:**
  - ECE 5984 - Advanced Linux Kernel Programming: **19563**

2 Instructor

Dr. Pierre Olivier
Postdoctoral Associate
ECE Dept., Virginia Tech
Office: 453 Durham Hall, Blacksburg, VA 24061

2.1 Instructor office hours information

- *Date and time*:
  - Monday 1 PM - 2 PM;
  - Wednesday 1 PM - 2 PM.
- *Location*: Durham 460.
- *Phone*: 540-231-2494
- *E-mail*: `polivier@vt.edu`

Additional office hours will be performed by the course TA, as indicated below.

3 Graduate Teaching Assistant (GTA) and GTA office hours

Fazla Mehrab
ECE Dept., Virginia Tech
Office: 450 Durham Hall, Blacksburg, VA 24061

GTA office hours:

- *Date and time*:
  - Tuesday 2:45 PM - 4:45 PM;
  - Thursday 2:45 PM - 4:45 PM.
- *Email*: `mehrab@vt.edu`
- *Location:* Durham 460.

4 Course objectives

The Linux kernel is one of the most, if not the most, advanced operating system kernels with wide acceptance in the industry and scientific worlds. It is used on a wide spectrum of computer hardware, from embedded and portable devices to servers and HPC platforms, also including regular desktop and laptop computers. Due to its diverse properties, many industrial projects are based on Linux (e.g., Google Android), and so is the majority of systems

software academic research. Knowledge about the Linux kernel internals, as well as kernel programming skills, are invaluable for a software engineer, especially one involved with systems software, but also for a hardware engineer to test new features or devices.

The course will teach Linux kernel programming following two complementary directions: (A) the study of the the different subsystems constituting the Linux kernel (roles, functions, and implementation); and (B) the development of kernel code in the form of new kernel modules as well as the modification of existing subsystems. Study and programming will concern all of Linux subsystems: processes, threads and scheduling; memory management; interrupt handling; device drivers; file system and block layer; network stack; synchronization and time management; boot process. In addition, specific techniques for programming in that particular environment that is the kernel will be studied, including kernel debugging methods, and large code base management and browsing tools. Notions of performance evaluation for the previously mentioned subsystems and their interaction with user application will be approached.

Upon completion of the course, the student will be able to:

- Identify the various subsystems composing the Linux kernel and describe their functionality, architecture, as well as the main characteristics of their implementation;
- Design, implement and modify Linux kernel code and modules for these subsystems;
- Test, debug and evaluate the performance of systems software in kernel or user space, using debugging, monitoring and tracing tools.

5 Prerequisites

- *4984*: ECE 4534 (Embedded Systems Design) ***or*** CS 3214 (Computer Systems)
- *5984*: Graduate standing
- *Both levels*: Good knowledge of C programming and the Linux command line is assumed. Knowledge concerning algorithms, data structures, and computer architecture is recommended.

6 Course meeting time and location

Meeting time: Tuesday and Thursday, 5 PM - 6:15 PM.
Location: Torgersen 1000.

7 Required and recommended texts

7.1 Required

- Love, R. (2010). *Linux Kernel Development, 3rd Edition*. Addison-Wesley Professional. Pp. xxv, 440.

7.2 Recommended:

- Bovet, D. P., & Cesati, M. (2005). *Understanding the Linux Kernel, 3rd Edition*. O'Reilly Media. Pp. xvi, 944;
- Corbet, J., Rubini, A., & Kroah-Hartman, G. (2005). *Linux Device Drivers, 3rd Edition*. O'Reilly Media. Pp xvii, 640;
- Mauerer, W. (2008). *Professional Linux Kernel Architecture, 1st Edition*. Wrox. Pp. xxx, 1368;
- Love, R. (2013). *Linux System Programming: Talking Directly to the Kernel and C Library, 2nd Edition*. O'Reilly Media. Pp. xx, 456.

8 Development environment

Some of the programming projects will require the student to install one or several Linux distributions (**natively**, for example in a *dual boot* way) on their personal machines. Moreover, the student personal machine should support hardware virtualization. Here are some resources to check for hardware virtualization support on one's personal machine:

- *Windows*:
  - Intel CPU: `https://downloadcenter.intel.com/download/7838/Intel-Processor-Identification-Utility-Windows-Version`

  - AMD CPU: `http://support.amd.com/en-us/search/utilities?k=virtualization`
- *Linux*: `http://www.cyberciti.biz/faq/linux-xen-vmware-kvm-intel-vt-amd-v-support`
- *Mac*: `http://kb.parallels.com/en/5653`

Should the above points concerning the development environment be an issue, the student is encouraged to contact the instructor.

9 Grading:

4984		5984	
Graded work	**Final grade %**	**Graded work**	**Final grade %**
4 small projects ($\sim$ 2 weeks of work each)	10-20 % each	3 small projects ($\sim$ 1-2 week of work each)	5-10 % each
1 large project ($\sim$ 4 weeks)	35 %	3 large projects ($\sim$ 3-4 weeks of work each)	20-30 % each
Final exam	10 %	Final exam	10 %

Most of the grade will come for programming projects. The final exam will be open-book.

10 Honor Code Policy

Adherence to Virginia Tech's honor code (Undergraduate Honor System: `http://www.honorsystem.vt.edu/`) is expected in all phases of this course. All graded work is expected to be the original work of the individual student unless otherwise directed by the instructor.

In working on the homeworks and projects, discussion and cooperative learning are allowed and, in fact, encouraged. However, copying or otherwise using another person's solutions to the homework/project problems is an honor code violation. Individual work is to be the work of the individual student. You may discuss general concepts, such as software libraries, Internet resources, or class and text topics, with others. However, any discussion or copying of homework/project solutions, specific code, or detailed report content is an honor code violation. All source material used in homework/project code and reports must be properly cited. If you are using a shared computer or disk, it is an honor code violation to leave your source, report, or other files on the computer where others may access them, and it is an honor code violation to access other students' files.

Please discuss any concerns about the honor code or any questions that you may have about what is or is not permitted with the instructor.

Any violations of the honor code will automatically be forwarded to the Office of the Honor System.

The Undergraduate Honor Code pledge that each member of the university community agrees to abide by states: "As a Hokie, I will conduct myself with honor and integrity at all times. I will not lie, cheat, or steal, nor will I accept the actions of those who do."

Students enrolled in this course are responsible for abiding by the Honor Code. A student who has doubts about how the Honor Code applies to any assignment is responsible for obtaining specific guidance from the course instructor before submitting the assignment for evaluation. Ignorance of the rules does not exclude any member of the University community from the requirements and expectations of the Honor Code. For additional information about the Honor Code, please visit: `http://www.honorsystem.vt.edu/`.

11 Special Needs or Circumstances

Any student with special needs or circumstances should feel free to meet with or otherwise contact the instructor.

- *Disability accommodation*: Reasonable accommodations are available for students who have documentation of a disability from a qualified professional. Students should work through Virginia Tech's Services for Students with Disabilities (SSD). Any student with accommodations through the SSD Office should contact the instructor during the first two weeks of the semester;
- *Religious accommodation*: If participation in some part of this class conflicts with your observation of specific religious holidays during the semester, please contact the instructor during the first two weeks of class to make alternative arrangements;
- *Accommodations for medical or personal/family emergencies*: If you miss class due to illness, especially in the case of an exam or some deadline, see a professional in Schiffert Health Center. If deemed appropriate, documen-

tation of your illness will be sent to the Dean's Office for distribution to the instructor. If you experience a personal or family emergency that necessitates missing class, contact the Dean of Students.

12 Course website

`http://www.ssrg.ece.vt.edu/lkp/`.

13 Course topic outline (tentative)

Topic	Req. Book Chapters
Generalities about Linux and software engineering techniques for large projects (version control, toolchains, configure, make, kernel installation, kernel code exploration/browsing)	1, 2, 5, 20
Kernel debugging techniques (printk, kernel traces/ftrace, kernel space debugging/gdb, QEMU/KVM debugging, gdb scripting)	18
Device drivers and data structures (generic device drivers, kernel basic data structures, kernel objects)	6, 17
Machine boot up process (BIOS, bootloader, compressed image, `start_kernel`, to multi-user/multi-task, user-space initialization)	-
Memory management (segmentation, paging, software MMU, physical memory space and allocation schemes, `get_free_page` / `kmalloc` / `vmalloc` / `cache_alloc` and NUMA affinity, process address space and `vma_struct`, `mm_struct`, remapping functions, I/O mapping)	12, 15
Linux process model and scheduling (kernel / user process context and task descriptor, kthreads, nptl, process and lightweight threading models, syscalls, kernel / libc / application interaction, scheduling and context switching)	3, 4
Kernel synchronization and time management	9, 10, 11
Interrupt / exception handling (interrupt paths, hardware and software interrupts, synchronous exception handling, signals, system call fast path)	7, 8
Virtual file system (block layer and block cache, file system driver, I/O scheduler, extended file system)	13, 14, 16