

Introduction

KDLP - Kernel Development Learning Pipeline MUNI Fall 2024

Rado Vrbovsky, Carlos Maiolino Vratislav Bendel, Izabela Bakollari



linkedin.com/company/red-hat



facebook.com/redhatinc



youtube.com/user/RedHatVideos





Overview

- Wellcome
- Course information
- KDLP
- Who are we
- The course
- Grades and expectations
- Assignments
- Suggested Literature



Course Information

- Lecture time: Mondays 14pm-16pm CET
 - Except 28th of October Public holidays
- Place: FI MUNI, Building S, Room S505
- Language: English
- Pre requisites
 - Clanguage knowledge
 - Basic knowledge about operating systems in general
 - Practical skills in Linux operating system (e.g. Fedora, Debian, Slackware)
 - Basic git
 - Own computer, preferably a laptop with x86 architecture
- Credits: The reward for passing the course is 3 credits.



Why is KDLP?

- Challenges in getting started with Linux kernel development without guidance
- The documentation gap and overwhelming terminology
- Navigating the kernel community and its strict rules
- Bridging the gap between academic learning and industry needs



What is KDLP exactly?

- Kernel Development Learning Pipeline
 - https://kdlp.underground.software/course/index.html
- The Kernel Development Learning Pipeline (KDLP) program aims to create a comprehensive Linux kernel talent pipeline to address the current shortage of qualified entry level kernel candidates.
- Created and developed in 2021 by Julia Denham, Joel Savitz and Charlie Mirabile at UMass Lowell in Boston
- Incremental Improvements based on student feedback (US course)
- Expansion to Israel in 2024 (Technion in Haifa)
- Expansion to Czech Republic in Fall 2024 (Course redesign for MUNI specifics) by Rado Vrbovsky, Izabela Bakolari, Carlos Maiolino, Vratislav Bendel and Leonardo Vaz



Rado Vrbovsky



- Kernel Maintainer for Red Hat since 2012
- Currently maintaining CentOS 9 and RHEL-9 kernels
- Project leader on the MUNI KDLP project
 - Turn your questions regarding organization to me
- Previously BIOS developer
- Geek, artist, manga reader

• Contact: rvrbovsk@redhat.com



Carlos Maiolino

- Red Hatter since 2008
- Filesystems engineer
- Upstream XFS maintainer
- Tech scuba diver and wall climber
- Contacts:
 - <u>cmaiolino@redhat.com</u>
 - <u>cem@kernel.org</u>
 - cem AT irc.oftc.net (yes, IRC still lives)



Vratislav Bendel

- Started as intern in Red Hat 2016
- Software Maintenance Engineer for RHEL kernel
- Focus on performance analysis & tuning
- Gamer at heart (video, board, card, .. :)
- <u>vbendel@redhat.com</u> (personal on demand ;)



The course - Linux Kernel is huge

functionalities lavers	human interfaces	system	processing	memory	storage	networking
user space interfaces system calls and system filos	HI char devices colev add nout handler vides tops vys cases to to vys gate	interfaces core System Call Interface Insubjaces, h insubjaces,	konel processes tolanoc c service sys fork sys exector sys for syn ydow sys exector sys syn ydow sys exector sys syn ydow sys anteroda syn ydow sys inter sys inter sys inter sys inter	memory access sys_bk sys_mmaster vm_oos sys_sterio sys_sterio sys_sterio sys_transt si_memnol (procineminite proclatilines may_enerry sys_mayor	b files & directories where the second seco	sockets access sys_sockets sys_socket sys_mont sock_ited
virtual	Insubsecurity security security security security security scole; create security inside; create security inside; create security inside; create serving inside in the security serving inside in the security inside security inside; create serving inside in the security inside security inside; create serving inside in the security inside security inside; create serving inside in the security inside security inside in the security inside security inside in the security inside security inside in the security inside in the security inside security inside in the security in the sec	Christer Mockel diversitiant bottyper bottyper bottyper diversitiant diversitta diversitiant diversitiant diversitiant diversitiant	threads schools set any man art york and any man work struct while a sut know thread causes thread the terms time	virtual memory vmafec_ine vmafloc_vmafloc vmafloc_vmap_anea vm_sizueri vm_sizueri vm_sizueri	View Jone Bostem View Jone Bostem View Jone With Mark View Bark View Bark Vi	address families int int sock main socket int order unit family, sos proto, ops int dram, ops
bridges cross-functional modules	debugging sys_stace log_bd registr kpobe profit handle_sysra kpb_bradipoit pert_event	diver, register device, diver pode toad medile module module gran kennel gazam	synchronization Michael Contention Michael C	minimati ger mage memory mapping knon dathe alloc mit inst mit stat mit stat	war jake war ja	socket file_ops socket splice sock sockappe up, sockappe
logical	HI subsystems ALSA V4L2 DRM video_device rousty/hefe	mil Supstan run bernoll Ebool, Briddown bibmanic statt ternoll do_inicials milet not nun_init_process	Scheduler kernelsched task strüct schedule timbout stag and schedule timbout schedule schedule schedule timbout schedule schedule	bigical memory physically margod memory and it profit part i part i	logical file systems ed.fe.gendes ed.fe.gendes	Protocols provident and a second provident and a sec
device control	abstract devices and HID class drivers driversinguid kbd (b. 05 driversignidm) mousedw driversignidm (b. 05 driversignidm) driversignidm	generic HW access request_moin pol_driver request_mem_region to make driver und driver und driver und to press und to pres	interrupts core requesting interstation interpreter state dotters state dot terms state dot terms state dot terms state state preters dot terms state state preters dot terms state dot terms state state state dot terms state state state state state	Page Allocator providention methods methods known (sath Jeff 1930) Jeff 1930 Jeff	block devices block gendisk block device operations mit geni see device see d	Teturente interfaces Teturente interfaces
hardware interfaces drivers. registers and interrupts	HI peripherals device drivers uc_drivers uga_cos usbé_dv iese_coor pombus	device access and bus drivers where the re- made where the re- where the re- where the re- made where the re	CPU specific sing and sate and the set of a set instruction of a set of a set system, and show mp PLATE states is and set of a set of a set system, and show mp PLATE states is and set of a set set of a set of a set of a set set of a set of a	physical memory control of the second archv80mm of the second second the archv80mm of the second the archv80mm of the second differences run physical differences run physical differences the second differences run physical differences run ph	storage drivers vitio_drivers drive_treve drivers innor drivers innor dr	network device drivers under pate pe200 pai pai pa e100 pai hano e100 pai hano e100 pai hano
electronics	user peripherals keyboard camera mouse graphics card audio	SDID UO more I/O PC/ I/O ports ACPI USB controller	CPU registers APIC interrupt registers APIC carteller		Storage controllers	network controllers





Assignments

Grades and Expectations

- There will be assignments (aka homework)
- Each assignment has a deadline when it must be delivered
- Each assignment is assigned a specific number of points based on its difficulty or importance
- These points add up to a maximum of 100
- To pass the course, students need to accumulate at least 65 points across all assignments



Assignments - Where, How?

- Assignments will be published over the course of time on the KDLP web page
 - https://fall2024-muni.kdlp.underground.software/index.html
- Each assignment will tell you what is exactly expected and what to deliver
- There is a private mailing list (you will not receive a copy of other students email)
 - kdlp-brno-assignments@redhat.com
- "Public" mailing list
 - kdlp-brno@redhat.com



Assignments

Dos and Don'ts

- Be creative!
- ASK!
- Meetings among students to work together are fine
- Exchanging ideas is fine
- Sharing code is NOT fine
- ChatGPT and alike are a powerful tool, please consider what you will learn if you will rely on them



Suggested literature

- Linux Kernel Development Robert Love
- Linux Device Drivers Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman
- The design of the Unix operating system Maurice Bach
- Linux Kernel Networking Implementation and Theory Rami Rosen
- LWN.net https://static.lwn.net/kerneldoc/index.html
- Unix Source Code https://github.com/lsahn-gh/unix-v6/tree/master/sys



Literature

Thank you!

Questions?

