MODULE 1

Computer Systems – Refresher

Prerequisite Modules: N/A

Sr. No.	Topic	Instruction Duration (hrs)	Lab Duration (hrs)	Difficulty Level (L, M, H)	Lab Outline	Comments
1	Physics of Electricity	0.5	0	L	N/A	
2	Power and Efficiency, ESD, EMC, EFT	0.5	0	L	N/A	
3	Analog and Digital Electronics	0.5	0.5	L → M	Generate Analog waveform using soundcard input and display using “Xoscope”. Generate Digital Waveforms using “JsigGen” and display using “Xoscope”.	
4	Number System - Bases	0.5	0.5	L	Practice lab on converting a String into base 2, 8, 10, 16. Signed Unsigned, and 2's complement	Binary, Octal, Hex, Decimal
5	Number System - Conversion	1	0.5	L	Convert from base 2 to 8, 2 to 10, 2 to 16, and Similarly between other bases.	
6	Floating-Point Arithmetic (IEEE 754)	0.5	1	L → M	Convert a decimal to floating point number i.e. Pi value in IEEE 754 format split in sign, exponent, and fraction, restore and estimate Precision loss	
7	Computer Architectures and Endianess, Micro-processor vs. Micro-controller	0.5	2	M	1) A script will be provided by instructor that prints two values on STDOUT against an input string: Checksum and CRC, paper exercise both Checksum and CRC. 2) Execute the script in step above on QEMU-X86 and QEMU-ARM, this will show the differences due to endianess	QEMU Lab (ARM, X86)
		4	**4.5**	**L = 4** **L → M = 2** **M = 1** **M → H = 0** **H = 0**		

MODULE 2

Linux Fundamentals

Prerequisite Modules: 1

Sr. No.	Topic	Instruction Duration (hrs)	Lab Duration (hrs)	Difficulty Level (L, M, H)	Lab Outline	Comments
1	Opensource: Philosophy, Licensing	0.5	0	L	N/A	Opensource vs. Closedsource, Licensing: GNU, MIT, FreeBSD, etc
2	Programming Fundamentals, Types of Programming Languages, Stability vs. Modernization, Compiled vs. Scripts/Interpreters	0.5	0	L	N/A	Procedural, Object Oriented, Hybrid, etc
3	Init	0.5	1	M	Src for a minimal bootable Linux OS will be provided by Instructor. Modify src for “init” to be placed under “/sbin” to print “Hello Techknox!”, then Compile and place “SystemV Init” under “/sbin” and configure.	QEMU
4	Bash Shell Basics: vim editor, Variables, Parameters, Special Characters, Customizing environment, X-Windows Display	1	1	L	1) Practice Lab on vim editor (insert, del, yank, del, etc), write a basic bash script To accept CLI parameters, interpret special characters. 2) Customize shell environment as per specs provided by instructor 3) Configure local X-Window server to tcp listen mode, display remote terminals On local display	
5	Bash: Regular Expressions (regexp)	0.5	1	L → M	Regular Expressions (RE) practice lab to pattern match using sh and grep, text searches and string manipulation using *, ^, $, ., escape sequences and Extended Res	
6	Bash: Tests, Operators, Comparisons, Loops, Arithmetic, Number Systems	1	1	L	1) Practice Lab on Tests (file/dir tests, conditional), Operators, Comparisons (strings, int), Loops (for, while, do while), Arithmetic, Number Systems, Using bc utility. 2) Write a script to generate Checksum and CRC for a string input on CLI	
7	Bash: Text File Processing	0.5	0.5	L	Practice Lab: Open, create, truncate, identify files, operations on files, search and replace, tabs and Spaces, locks, head and tail, stats, cut and paste, join, merge	
8	Bash: IPC, I/O Redirection, Pipes, FIFO, File I/O	1	1	L → M	Create three files – PIPE and its file handle, reader and writer, then produce And consume. Redirect STDIN to file handle for PIPE. Practice Redirection in Various scenarios.	
9	Users, Groups and Permissions	0.5	0.5	L	Practice Lab: Add and modify user and groups, permissions on files and folders Useradd, Usermod, Chown, Chmod, Setgid, Setuid, Sticky Bits,	
10	Process Scheduling and Management	0.5	0.5	L → M	Write a shell script that runs infinitely. Schedule it to run in the background at Specified days and times. Watch the process for failure. Another process that Kills the infinite process, the infinite process should then restart and notify.	
11	Filesystem Hierarchy and Management	0.5	0.5	M	Debug inodes of filesystem, hard and soft links, use of stat utility	Inodes, Links, VFS Layer, fsck and fstab
12	/dev and udev, /sys, and /proc filesystems	1	1	L	1) Display process stats from /proc<pid>, process written in #10. Modify process Priority and parameters in /proc fs. Use of 'stat' command and inodes. 2) Probe a device using udev, make udev rules for a usb device	
13	Utilities, Bash Network I/O	2	1	L	1) Practice Lab: Man, cat, mkfifo, jobs, kill, renice, stty, ulimit, exec, netcat, lsof, lslk, socat, fuser, Time, top, pstree, etc 2) Client and Server in bash, /dev/udp, /dev/tcp	
14	Kernel: Intro, configure, compile, optimize	1	2	L → M	1) Walk through the kernel build and configuration system 2) Download kernel src, configure, understand and modify sources, compile, install, Initialize	
15	Systems Administration: X-Window Display System	0.5	0.5	L	Initialize X-server over TCP, client server display, set xterm params, switch, destroy	
16	Systems Administration: Run Levels and Init Scripts	1	1	L	Create run level scripts for run-level 3, install for init, initialize, verify	
17	Systems Administration: Date and Time, Task Automation	0.5	1	L	set system date time tz, automate scripts at a schedule to run once and Periodically	
18	Systems Administration: Debugging	2	2	M → H	1) Debug a process provided by instructor - memory, I/O, CPU, Network, core, /proc, Using utilities for each sub-system i.e. ss, lsof, perf, itop, tcpdump, ltrace, strace Pidstat, lttng, iotop, ethtool, vmstat, sar, gdb (cores) 2) Run two processes: P1 by student and P2 by instructor (P2 not known by student), P1 is multi-threaded and critical – it must survive, P2 hogs memory and forces P1 to die. OOM_KILLER starts killing P1 threads, adjust OOM SCORES and achieve the goal.	Profiling, Core Dump and Analysis
19	Systems Administration: System Security	2	2	L → M	1) Create AppArmour profile for a process/executable: gen profile using aa-genprof, train using complain mode, apply enforce mode, test and verify 2) Create a One Time PassWord (OTPW) script to generate random passwords with entropy from /dev/[u]random and attempt a successful sign-in 3) set suid/sgid bits on filesystem and analyze effects	AppArmor, Log Analysis, /dev/[u]random (randomness, entropy), suid/sgid
20	Systems Administration: Remote Access and Control	1	1	L	Setup restricted remote access for ssh and telnet, using wireshark see differences Between ssh and telnet, use wrappers i.e. stunnel for securing telnet	ssh, telnet
21	Systems Administration: Network Configuration	0.5	1	L	Configure network interfaces for static and dhcp, bridge interfaces, interface aliases, Promisc mode, use Utilities ifconfig, ethtool, ip, openvswitch, br	
22	Systems Administration: Network Monitoring and Management	1	1	M	1)Monitor network interface statuses, configure and install mrtg and nagios, monit 2) Write a script to monitor a process, alert using smtp, restart it	Nagios, Process Monitoring, SNMP
23	System Performance and Tuning, Optimization	1	1	M → H	1) Tune a process for priority, various parameters in /proc fs 2) Tune system wide using sysctl, /proc, and sys – Memory, Network, and I/O Sub-systems	
		20.5	**21.5**	**L = 13 L → M = 5 M = 3 M → H = 2 H = 0**		

MODULE 3

Python Fundamentals

Prerequisite Modules: 1, 2

Sr. No.	Topic	Instruction Duration (hrs)	Lab Duration (hrs)	Project Duration (hrs)	Difficulty Level (L, M, H)	Lab Outline	Comments
1	Introduction and Basics	0.5	0	0	L	PROJECT: All Submissions will be through CI Toolchain	
2	Object Orientation	2	2	0	L	Instructor will provide code skeleton: 1) Write a program to carry out three tasks, First two tasks are independent and third task depends on first task, write procedural code. 2) Modify the program to have the three tasks in classes and in independent files, rewrite code structure to optimize for OOP and Security.	Classes/Objects, Polymorphism, Inheritance, Instantiation, Singleton
3	Data Types and Structures	1	1	0	L	1) Write a program to keep reading data from CLI and store in tuples, it Should read three commands write, update and delete. 2) Modify code to use Lists instead of Tuples 3) Modify code to use Dictionary instead of Tuples 4) Modify code to use nested structures i.e. Lists and Dictionaries inside Dictionary	
4	Introduction to IDLE, vim	0.5	0.5	0	L	Write and execute snippets of program written in #3 on Idle shell, test and debug. Instructor will provide test cases.	
5	Libraries and usage	2	2	2	L	Modify code in #3 to include functions from python libraries: string, os, Time, shutil, json, etc	pydoc server, import, from import, json
6	Interaction with bash	0.5	0.5	4	L	1) Modify code in #5 to perform sysadmin tasks, include a sysadmin class, Use “os” and “subprocess” libraries 2) Modify the code in previous step to use “shutil” and “bash” libraries Instead	
7	Regular expression and pattern matching	0.5	0.5	1	L	Modify code in #6 to add a Filter function that filters CLI parameters for valid Data using the RE library	
8	File I/O	0.5	0.5	4	L	Modify code in #7 to read data from CLI and write to Files after Filtering, add Data CRC to another file, add a function to randomly modify those files to Corrupt data, another function should then read data and correct errors with CRC technique. Use “crc16” library	Error Detect and Correct
9	Queues	1	1	2	M	1) Modify code in #8 to use Queues instead of files, the serially executing functions should read and write to Queues. Use the “multiprocessing” library 2) Modify the code in step above to use AMQP instead of Multiprocessing, use RabbitMQ as an external queue	Multiprocessing, Pika (RabbitMQ)
10	Multi-threading	1	1	8	M	1) Modify code in #9 to use Threads instead of serially executing functions, Threads should be in independent classes, use “Threading” library 2) Note and List any unpredictable read/write behavior by Threads	
11	Semaphores, Mutex, Locking	0.5	0.5	4	M → H	Modify code in #10 to “acquire” and “release” locks on Queues, then the Threads should notify and join the dependent threads, use “Threading” library	
12	Hardware I/O	1	2	6	M → H	1) Create emulated HW Serial Port using “socat” 2) Modify code in #11 to read from Serial Port created in step above, instead Of CLI 3) Add a Thread to output status to Serial Port	UART/RS-232
13	Configuration Files	0.5	0.5	4	L	1) Add configuration for Queue (RabbitMQ) and Serial Port in a config file, use the format specified by “configparser” library 2) Modify the code in #12 to read Queue and Serial Port configuration Parameters from the config file created in step above	configparser
14	Daemons	0.5	0.5	1	M	The program written in #13 runs from CLI in the foreground, daemonize it Using “python-daemon” library	
15	Signals and Timers	1	1	1	M	1) Modify the code in #14 to read signals on CLI, add a Thread for this. A signal Handler should be defined that also detects the CTRL-C SIGNIT and Ignore it. Use “signal” module 2) Add a timed Signal as Alarm 3) Modify code to expire a Thread, use “timeit” module 4) Modify code to schedule execution of a thread, use “sched” module	
16	Virtualization I/O	0.5	1	1	L → M	1) Modify code in #15 to schedule execution of a KVM VM, carry on a task inside VM and destroy the VM, task inside VM must complete successfully 2) Modify task inside VM to infinite execution time, time the VM so it is destroyed After a certain threshold	libVirt
17	Python module extensions in C	0.5	1	0	H	Instructor will provide C code for UDP client 1) Add a python module as wrapper to C code 2) Use “import” the python module - C code	python.h
18	Debugging and Analysis of Python Code	0.5	0.5	0	M	The lab is to find out: 1) Identify fast or slow execution of program 2) Identify speed bottlenecks 3) Find memory usage 4) Find memory leaking Use modules “line_profiler”, “memory_profiler”, “objgraph”, “pdb”	
		14.5	**16**	**38**	**L = 9 L → M = 1 M = 5 M → H = 2 H = 1**		

MODULE 4

Inter-Networking with Python and TCPIP

Prerequisite Modules: 1, 2, 3

Sr. No.	Topic	Instruction Duration (hrs)	Lab Duration (hrs)	Project	Difficulty Level (L, M, H)	Lab Outline	Comments
1	TCP/IP Protocol Fundamentals	0.5	0	0	L	N/A	Layers 1 – 7
2	TCP/IP: LAYER-1	2	0	0	M → H	N/A	ETHERNET/802.3, WiFi/802.11, Networking Fabric
3	TCP/IP: LAYER-2	1	0.5	0	M	Wireshark and “tcpdump” Lab on a VM, put an interface in promisc Mode and attach to a sniffer, filter L2 802.3, 802.1Q frames	MAC, VLANs, Sniffer Lab
4	TCP/IP: LAYER-3: IPv4	4	4	2	M	1) Paper calculations on fragmentation 2) Divide an IPv4 network in subnets with variable number of Hosts for company departments QA, HR, IT, SW, HW, FA, HSE, EXCOM 3) Write a python program that outputs a subnet against input Of 'no. of hosts', print to STDOUT the 'gateway', 'network ID', and 'subnet mask' 4) Design a DMZ with local and public DMZ 5) Define routing tables 6) Virtual Routing tables for a router with 3 NICs, use “iproute2” Router HW with required NICs provided by Instructor	IP, Fragmentation, Forwarding, Routing – Static, VR, VLSM
5	TCP/IP: LAYER-3: IPv6 basics	1.5	1	0	M	Modify the lab in #4 with IPv6	Instructor: Ibrahim
6	TCP/IP: LAYER-4	3	3	6	M	1) Write a UDP client and server, modify program in Module 3, Use “SocketServer” module 2) Write a TCP client and server, modify program in step above, use “SocketServer” module 3) Write a minimal python “ping” utility suing raw sockets 4) Write CLI based UDP client and server using “netcat” and “socat” for messaging 5) Write CLI based TCP client and server using “netcat” and “socat” for file sharing	ICMP, TCP, UDP
7	TCP/IP: LAYER-5	0.5	1	4	L → M	1) Write a web server (http) in python, without using any Module, use “SocketServer” module only, headers to comply with RFC 7231 – HTTP/1.1 (Semantics and Content) 2) Write a web server (http) in python, use “SimpleHTTPServer” Module	HTTP
8	TCP Protocol Implementations and Algorithms	0.5	0.5	0	M	Configure Linux TCP stack to use various congestion control Algorithms on run-time using /proc fs or sysctl, benchmark effects On large file transfers by injecting congestion	Cubic, Hybla, Reno, CoV effects, etc
9	Delays, RTT, Latency, Jitter, Packet Loss	1	1	0	M	Use kernel “netem” module to inject delay and packet loss, analyze Performance of algorithms from #7	
10	Masquerading/NAT, Firewalls and Security	4	4	0	M	1) Port and protocol scan a vulnerable host using “nmap” 2) Secure the network designed in #4, list of ports, services and Requirements will be provided by instructor	IPTables, Snort, nmap
11	QoS	2	2	0	M → H	1) Limit bandwidth used over TCP and UDP in a network segment, For network designed in #4 2) Calculate BDP	Queuing Disciplines/qdisc
12	Networking Services – DNS, HTTP, SSH	1	2	0	L → M	1) Deploy and configure a SOA DNS Server for a dns zone, design For network developed in #4 2) Deploy and configure a secondary DNS Server, and enable zone Transfers between both 3) Write python program for ssh client using “paramiko” module	DNS, HTTP
13	Regulatory Authorities: IETF, IANA, ISC, InterNIC	0.5	0.5	0	L	Match Linux network service port/service pairs and DNS records with Authority records	RFC
		21.5	**19.5**	**12**	**L = 2 L → M = 2 M = 7 M → H = 2 H = 0**		

MODULE 5

C Development on Linux

Prerequisite Modules: 1, 2

Sr. No.	Topic	Instruction Duration (hrs)	Lab Duration (hrs)	Difficulty Level (L, M, H)	Lab Outline	Comments
1	C Programming and Compilers Intro	1	0	L	N/A	
2	GCC Compiler Suite Fundamentals	1	1	M	1) Install and configure gcc, gcc spec, discet gcc suite to Understand its external components i.e. ar, as, nm, objdump, ld, cc. 2) Compile a test src file. Compile on different architectures and analyze asm Code 3) Learn GCC extensions to C language	ar, as, nm, objdump, ld, cc, etc
3	C Programming Coding Standards	0.5	0.5	L	Practice Lab: apply coding stnadrads formatting to a src file, use Lint	GNU C, ANSI C, C99
4	Stages of Compilation	0.5	1	L → M	Compile a src file and breakdown in stages of compilation using Individual gcc components, inspect files generated in each stage	preprocess, asm, object code, link
5	gLibc and variants	1	1	M	1) Compile using Glibc and a variant, use flags i.e. LDFLAGS, CFLAGS 2) Static and dynamic link a src file 3) Analyze static and dynamically generated binary, use of 'ldd'	static and dynamic linking
6	Data Types	1	1	L	Write code for a program that prints messages periodically Messages are both signed and unsigned, float and int, messages are Sent as a frame of bytes aligned	CPU Architecture dependencies, int, float, Un|signed
7	Variables, constants, and storage types	0.25	0.25	L	Modify the program in #6 to store variables in an external header file, Reference variables using extern, use of static variables	static, volatile, extern, etc
8	Operators	1	1	L	Modify the program in #7 to perform various operations on data, at Run-time and compile time, perform bitwise and logical operations	Arithmetic, Relational, Logical, Increment and Decrement, Bitwise, Assignment, and Conditional
9	Operator Precedence and Order of Evaluation	0.25	0.25	L	Practice Lab to experiment with the topic	
10	Type Conversions and Casting	0.25	0.25	L	Practice Lab to experiment with the topic, typecast to 'void', precision loss	
11	Control Flow and Conditional Statements	0.25	0.25	L	Practice Lab to experiment with the topic	
12	Standard I/O and Formatting	0.5	0.5	L	Practice Lab to experiment with the topic	
13	Functions and prototypes	0.25	0.25	L	Modify the program in #9 to breakdown code in functions, define their Prototypes	
14	Strings and Characters	0.5	0.5	L	Modify the program in #13 to exercise strings and characters, Manipulate strings and use c library string.h	pragmas
15	File I/O	0.5	0.5	L	Modify the program in #14 to write message to two separate text files Read from files, seek in files	
16	Command-line Processing	0.5	0.5	L	Modify the program in #15 to read from the CLI using glibc getopt	getopts
17	Arrays	0.5	0.5	L	Modify the program in #16 to read strings and int from the CLI and Store in 1D and 2D array, then read from those arrays	
18	Pointers	2	2	M → H	Modify the program in #17 to use Pointers, string manipulation using Pointers i.e. reverse a string, perform pointer arithmetic, char substitution Using pointers	void
19	Structures, Union and Objects	1	1	M → H	Modify the program in #18 to create Objects using pointers and Structures	
20	Bit Fields and Bitwise Operations	0.5	1	M → H	Modify the program in #19 to add functionality for embedded code, Optimize memory usage using bit fields, verify using sizeof(), perform Multiply and divide operations on integers using bitwise operators	
21	Function Pointers	0.5	0.5	L	Modify the program in #20 to add function pointers and call functions Using pointers	
22	Header Files, Extern and Reusable Code	1	1	M → H	Modify the program in #21 to split the code in more than one header File, use external variables and functions, build library from code	
23	Preprocessor Directives and Macros	1	1	M	Modify the program in #22 to split code in object-like and function-like Macros, use of poison, dependency, warning, error, pragmas,	
24	Data Structures	1	1	M	1) Modify the program in #23 to implement a singly linked list 2) Modify the same program to use a Queue instead	Linked Lists, Stacks, Queues
25	Build Tools	0.5	1	M	1) Construct a Makefile to compile the program developed in #24, compile Using 'make' instead of gcc command-line 2) Construct a Scorn file to compile the program developed in #24, compile Using 'scorn' instead of gcc command-line	
26	Memory Allocation and Management	0.5	0.5	M	Modify the program in #25 to allocate dynamic memory to Objects/structs, Memory map the files and benchmark run-time	malloc, calloc, mmap
27	Algorithm Complexity, Code Optimization and GCC	1	1	M → H	Optimize the code written in #26 using gcc parameters and attributes, Optimize source for function growth and optimized run-time	O(linear), O(log), O(1), gcc attributes
28	Debugging with GDB	0.5	0.5	M → H	Instructor will provide a modified program from #27 to SEGFAULT and dump core, inspect the core with gdb, find problem, and fix	Profiling, Core Dump and Analysis, Attach to process
29	Memory Debugging with Valgrind	0.5	0.5	M → H	Dynamically allocate memory in multiple context, and ensure that program In #28 does not have memory leaks or violations	
		19.75	**20.25**	**L = 15** **L → M = 1** **M = 6** **M → H = 7** **H = 0**		

MODULE 6

Linux Internals and Development

Prerequisite Modules: 1, 2, 3, 4, 5

Sr. No.	Topic	Instruction Duration (hrs)	Lab Duration (hrs)	Difficulty Level (L, M, H)	Lab Outline	Comments
1	Architecture	1	1	H	Source for custom Linux minimal distro will be Provided by instructor. 1) Write script to generate a custom file system 2) Kernel space: compile kernel from source and Install binaries and headers in appropriate locations in the target filesystem 3) Userspace: Compile binutils, inetutils, etc from sources and install in appropriate locations in the filesystem 4) Compile uBoot and install on target disk 5) Boot the system in QEMU and verify	filesystem, binutils, kernel and user spaces
2	Boot Process	0.5	1	H	Bootloader Lab, uBoot on QEMU, first and Second stage bootloading process, modify Uboot source to add custom code, write to MBR And reboot system to verify	Bootstrapping to Init
3	Processes	1	1	H	Create a custom init process and fork child Processes, install and boot under the custom Linux distro	fork, exec, vfork, clone
4	Daemons	0.5	1	H	1) Modify the code in #3 to create a daemon process That detaches from terminal and serves in the Background 2) fork child processes, and use wait, exit and atexit system calls to notify parent 3) Learn parent and child relationship	
5	Virtual Memory Sub-system	1	1	H	1) Learn the internals of VMM by using /proc and Vmstat 2) View the page table entries in running kernel 3) Modify slab allocator and MMU parameters 4) Use HugeTLBfs with large data and debug	
6	Processes and Threads	2	2	H	1) Modify code in #4 to convert processes to Threads using pthread library 2) Program to yield the processor using sched.h 3) Prioritize processes using getpriority() and Setpriority() calls 4) Two threads should read write to a queue Randomly, use /dev/(u)random 5) Use semaphores to prevent deadlocks	pThreads, semaphores, mutex, prctl
7	Inter-process Communication (IPC)	1	1	H	Modify code in #5 to implement named pipes And message queues, thread should use both For read and write	
8	Shared Memory	1	1	H	1) Modify code in #7 to execute external Processes 2) The external processes should read write to A shared memory (SHM) and prevent deadlock	
9	System Calls	0.5	0.5	H	1) Implement a custom System Call 2) System Call tracing	fork, exec, vfork, clone, ioctl, strace, syscall.h
10	I/O Sub-system	1	1	H	1) Experiment with the I/O Scheduler, use of CFQ And I/O priorities 2) Modify code in #8 to use SCHED_RR and SCHED_FIFO priorities and note behavior	
11	Networking Sub-system	1	1	H	1) Debug and tune network device buffers, ring Buffers 2) Benchmark using ss, iperf, and apache-benchmark 3) Write code to create hooks into Ethernet device	
12	Sockets	1	2	H	1) Raw sockets lab, modify code in #10 to implement L2 to L4 headers only, two processes should Exchange data via net dummy device and Print out at each layer 2) Transmit raw frames between two computers And display stack data using Wireshark 3) Write a UCP client and server in C and debug Sockets using ss and /proc, monitor socket states	UDP, TCP, RAW
13	Scheduling Sub-system	1	1	H	1) Schedule process in #12 using schedule() in sched.h 2) Debug process states from /proc 3) Use various scheduling algorithms provided by the Kernel	process scheduling
14	File I/O, File Systems Brief	1	1	H	1) Modify code in #13 to use dlopen() and dlsym() from Dynamic Loading (DL) API 2) Debug ELF Headers, use of "readelf" utility	*ELF, ext2, ext4, journaling, mmap*
15	Signals, Timers	1	1	H	1) Modify code in #14 to implement various signals Defined in signal.h 2) Define a custom signal and signal handler, use in Code from step above 3) Modify code from step above to use a signal-safe, That is reentrant 4) Block a signal and retrieve a signal	
16	Interrupts, polling, NUMA	1	1	H	1) define a custom soft IRQ 2) Modify code in #15 to act on soft IRQ defined in Step above 3) Balance interrupts using "numactl" utility 4) Debug interrupts using /proc	
17	Kernel Modules	1	1	H	1) Write a custom kernel module 2) Insert in the running Kernel 3) Remove the module	
18	Profiling	1	1	H	1) Simulate a bottlenck in filesystem layer – instructor Provides code 2) Use "opcontrol" to use OProfile to collect samples During GLOBAL_POWER_EVENTS 3) Start "opcontrol" and stress test filesystem 4) Dump "opcontrol" data for analysis	
19	Tracing	1	1	H	Use Linux Trace Toolkit to create and analyze LTT Events, use trace.h	
20	Tools & Techniques for Diagnostics	1	2	H	1) Learn Debugging SEGFAULT's and Core Dump's With GDB 2) Create a process to raise a SEGFAULT and dump Core 3) Analyze core dump with GDB 4) Fix code and attach gdb to running process, force Process to core dump and analyze 5) Debug usb devices	
		19.5	**22.5**	**L = 0 L → M = 0 M = 0 M → H = 0 H = 20**		

MODULE 7

Python Commercial Applications

Prerequisite Modules: 1, 2, 3, 4

Sr. No.	Topic	Instruction Duration (hrs)	Lab Duration (hrs)	Project Duration (hrs)	Difficulty Level (L, M, H)	Lab Outline	Comments
1	Plotting and Graphing	1	1	0	L → M	Read network card interrupt stats From /proc and plot using matplotlib	matplotlib, gnuplot
2	GUI – wxpython	2	2	0	L → M	1) Write a GUI for a UDP client and Server chat client 2) Data should be stored in pickledb	
3	Design RESTFUL Api with Eve	1	1	0	L → M		
4							
5							
6							
7							
8							
9							
10							
		4	**4**	**0**	**L = 0** **L → M = 3** **M = 0** **M → H = 0** **H = 0**		

MODULE 8

Enterprise SDN and NFV – Engineer Track

Prerequisite Modules: 2, 3, 4, OpenStack

Sr. No.	Topic	Label	Instruction Duration (hrs)	Lab Duration (hrs)	Difficulty Level (L, M, H)	Day	Hours/day	Comments
1	Introduction to SDN and NFV Training Module	REFRESHER	0.25	0	L → M	1	1.25	
2	Linux Architecture and Networking sub-system		1	0	L → M			
14	Introduction to: * Software Defined Networking (SDN) * Network Function Virtualization (NFV)	MAIN COURSE	1	0	L → M			* What are SDN and NFV, and relationship * SDN and NFV – physical to virtual * Motivation behind deploying NFV * NFV requirements, benefits and Challenges
15	Virtualization Fundamentals		0.5	0.5	L → M			* Hardware and OS Virtualization * Hypervisors: QEMU, KVM * Benefits: fault-isolation, avoid vendor lock-in, Scalability, standardization, high-availability, etc * Proprietary vs Open-source * Building Blocks
16	OpenStack Overview		0.5	0	M			
17	Introduction to OpenFlow		1	0.5	M			
18								
19	Compute Virtualization		0.5	1	L → M			
20	Storage Virtualization		0.25	0	L → M			
21	Network Virtualization		1	1	L → M			
22	Network Virtualization: Switching and Bridging		1	0.25	M	4	8	OpenVswitch
23	Network Virtualization: IP Routing		1	0.25	M			iproute2, ip
24	Network Virtualization: QoS		0.5	0	M			
25	Network Virtualization: Firewalling – IDS + IPS		1	0	M			
26	**Lab Exercise 1: Enterprise NFV**		0	4	M → H			
27	Network Virtualization: MPLS VPN		1	0	M → H	5	8	
28	Lab Exercise 2: Enterprise MPLS VPN		0	2	M → H			
29	Network Virtualization: Dynamic IP Routing with BGP		1.5	0	M → H			Quagga
30	Network Virtualization: Dynamic IP Routing with OSPF		0.5	0	M → H			Quagga
31	Lab Exercise 3: Service Provider		0	3	M → H			
			12.5	**12.5**	**L = 0 L → M = 7 M = 6 M → H = 6 H = 0**			

MODULE 9

SDN and NFV – Developer Track

Prerequisite Modules: 2, 3, 4, 5, 8

Sr. No.	Topic	Instruction Duration (hrs)	Lab Duration (hrs)	Difficulty Level (L, M, H)	Day	Comments
1	Introduction to RYU	0.5	0			
2	RYU Packet Library	1	1			
3	Introduction to OpenFlow + RYU	1	0.5			
4	Opendaylight RESTFUL Api	2	2			
5	Neutron Python Api	2	2			
6	Introduction to opNFV	1	1			
7	opNFV Internals	2	2			
8						
9						
10						
		9.5	**8.5**	**L = 0** **L → M = 0** **M = 0** **M → H = 0** **H = 0**		

SUMMARY

Training Modules

Sr. No.	Module	Phase (B)Baseline (I)ndustrial	Instruction Duration (hrs)	Lab Duration (hrs)	Project Duration (hrs)	Total Duration (hrs)	Average Difficulty Level (L, M, H)
1	Computer Systems – Refresher	B	4	4.5	0	8.5	L
2	Linux Fundamentals	B	20.5	21.5	0	42	M → H
3	Python Fundamentals	B	14.5	16	38	68.5	L → M
4	Inter-Networking with Python and TCPIP	B	21.5	19.5	12	53	M
5	C Development on Linux	I	19.75	20.25	0	40	M → H
6	Linux Internals and Development	I	19.5	22.5	0	42	H
7	Python Commercial Applications	I	0	0	0	0	M → H
8	SDN and NFV - Engineer Track	I	12.5	12.5	0	25	H
9	SDN and NFV – Developer Track	I	9.5	8.5	0	18	H
			121.75	**125.25**	**50**	**297**	M → H