Linux + Certification

Introduction:

* LXO-103 == LPIC 101-400
* LXO-104 ==LPIC 102-400
* 500/800
* 60 questions
* 90 minutes and is valid for 5 years
* $200 to sit for exam.
* Watch video
* Take notes
* Re-watch video and pause at each step
* Read documentation for anything you don’t understand

**About Linux and Installation:**

What is linux?

* Unix-like computer OS for free and open source distribution.
* Originally made for x68 architecture
* Largest install base now with general purpose OS
* Some want to name it GNU/Linux or Linux
* Components
  + **Boot loader** – software that manages boot process
  + **Kernel** – core of OS. Manages CPU, memory
  + **Daemons** – processes that start during the boot that support the system
  + **The Shell** – access to the CLi and control the whole system.
  + **Graphical Server** – X server, sub system which displays graphics
  + **Desktop environment** – what the user interacts with.
  + Applications
* Free
* Stable
* Secure
* Open Source

Linux Distribution

* Collection of software
* **Package management system** – help install, upgrade, and remove software
* Keeps your server up to date
* Redhat
  + CentOS
  + Fedora
* Debian
  + Ubuntu
  + Mint
* SuSE
* Gentoo
* Arch
* Slackwareß

How can we run linux?

* Your own PC
* Someone else’s PC, Shared hosting, Cloud Provider (AWS, GCP, Azure)
* Virtualization

Installing Ubuntu & CentOS

* Not good to put a boot loader in the master boot record. Okay on VM.
* Enabling bridge network mode
  + CentOS
    - Set adapter to bridged
    - Cd /etc/sysconfig/network-scripts
    - Sudo vi ifcfg-enp0s3 (a=append)(:wq) (ESC)
    - Onboot = yes
    - Ip addr is to see IP address
  + Ubuntu
    - Just set adapter to bridged
* Sudo loadkeys us – shows keys local to the region.

**System Architecture:**

Boot the System

* **\*BIOS (Basic Input Output System**) – firmware that provides hardware initialization of the boot up of your system.
* **MIT (Motherboard Intelligent Tweaker)** – allows you to provide tweaks on the CPU and overclocking.
* **System Settings** – Time and Date
* **Boot Option priorities** – let us select which devices are booted from
  + **UEFI (Unified Extensive Firmware Intelligence)**– UEFI -> EFI Boot Loader -> Kernel -> Operating System
    - Has its own file system
    - Each should have its own boot loader
  + **Legacy booting** – BIOS -> MBR(Master Boot Record) -> BootLoader -> Kernel -> OS
* **MBR** – identifies where and why a OS is located to boot from.
* **Kernel** – lowest level of replicable software that interacts with the hardware.
* **Ps aux | head** – goes over the processes listed
* **Sysvinit** – first commercial version of linux available
* **Systemd** – low memory and perform’s better than sysvinit. Easier to read than sysvinit
* **Upstart** – Ubuntu developers made by it.
* **Dmesg/dmesg -T** – shows kernel messages when the system was booted

Determine and configure hardware settings

* **Udev** – device manager for the kernel of your system
  + Low level access to the linux device tree
  + Handles user space events
    - Loading firmware
  + Provided by a temporary filesystem (tmpfs)
  + Mounted to /dev on startup
* Dbus – inter-process communication mechanism that allows processes to talk to each other
  + Provides a high-level object oriented programming interface
* Sysfs – virtual filesystem that presents varios information about kernel subsystems
  + Mounted to /sys
  + Hardware devices
  + Drivers
* Procfs – similar to sysfs but presents information about processes and information about system information.
  + Mounted to /proc
  + Can be used to interface with the kernel and change parameters on the fly
  + **Cat cmdline**
  + **Lsmod | less –** page by page version of module
  + **Modprobe –** enable a mod
    - **Rmmod –** remove a mode
  + **Lspci** – shows devices connected to the computer

Run levels, boot targets and how to shutdown and reboot system

* **Run level (inittab)** – number between 1-9 and determines which scripts are used based on the run level
  + 0 – halt or shut down
  + 1 – single user mode
  + 2 – multi-user mode without networking
  + 3 – normal boot
  + 4 – Unused/customizable
  + 5 – Run level 3 + GUI display manager
  + 6 – reboot
  + rcS.d – is run level used during every level
  + **init telinit** – used to change the run level
  + **man telinit** – manual command
  + **wall** – writes to the shell as to everyone that is logged in.
  + **systemctl** – is a way to control systemd devices or targets
    - can be used like init telinit
* **Scripts** – little programs held on the system that can be used.
* System run level
  + /etc/system/system
  + Package scripts
    - /user/lib/systemd/system
  + System packages take power over packaged scripts
  + **Targets** – are like run levels. Named instead of numbers

**Linux Installation and Package Management:**

Design a hard disk layout

* /usr – user binaries where packages are installed
* /home – user files
* /boot – where the boot loader is found
* /etc
* /var – variable files
* /tmp – unique storage area where everyone can write too
* Partitioning
  + Separation of files
  + Allows dual booting
  + Divide storage into multiple pieces
  + Data organizations
  + System protection
  + Helps ensure data doesn’t overflow into other directories.
  + Swap partition can be used to free up more memory
  + **LVM** – Logical Volume Manager – split physical partitions into pools

Install a boot manager

* Common boot loaders
  + LILO
  + GRUB Legacy
  + GRUB 2 **(grub-probe –version)(grub-install –version)**
    - **Less grub.cfg command**
    - **Sudo vi /etc/config/grub.d** 
      * need to run sudo **update-grub** to update the changes made.
      * Quiet mode will eliminate a lot of messages during bootup
    - Find where a script is located – **which update-grub**
    - Can have systemd boot from another target – like **vg-root ro systemd.unit=rescue.target**
    - **Sudo grub-install /dev/sda**

Manage Shared libraries

* Shared reuseable pieces of code
* **Static linking** – application included in a library
* **Dynamic linking** – two applications use one single library
* **/etc/ld.so.conf** is where the shared libraries would be
* **libc.conf** is in C programming language
* **man ldconfig** does the dynamic linking in a directory
* **environment variables –** customized pieces of text that are specific to your shell
  + **export LD\_LIBRARY\_PATH=/home/nick/lib/**
  + **echo $LD\_LIBRARY\_PATH**
  + **LDD –** prints shared object dependencies

Debian package management

* Dpkg
  + Debian Package Manager
  + Install software, ugrade, remove, low-level tool
  + **Dpkg –l | less**
  + **Wget to pull from a url**
  + **Dpkg –purge dlocate**
  + **Doesn’t find dependencies and doesn’t automatically install them**
* Apt
  + Advanced packaging Tool
  + High level tool
  + Install, upgrade, remove packages, upgrade entire system and handles package dependencies.
  + Uses online repositories
  + Installs dependencies.
  + **Apt-cache depends apache2 | less** – search for packages and looks at dependencies of a package.
  + **Apt-cache search nginx**
  + **Apt-get upgrade** – can’t upgrade all packages
  + **Apt-get dist-upgrade** – upgrades everything and deletes everything that it won’t use
  + **Sudo apt autoremove**