LPIC-1 Exam Workbook

A Chapter-by-Chapter Syllabus with Practice Questions

Version 1.0

Author: Mehdi JAFARIZADEH
Date: January 1, 2025

Contents

Chapter 1

Topic 101: System Architecture

101.1 Determine and Configure Hardware Settings

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 101.1**
- **Weight:** 2

Key Knowledge Areas

- Enabling/disabling integrated peripherals (BIOS/UEFI).
- Identifying different types of mass storage devices.
- Determining hardware resources for devices (IRQ, DMA, etc.).
- Using tools (`lsusb`, `lspci`, `lsmod`) for hardware inspection.
- Manipulating USB devices.
- Understanding `sysfs`, `udev`, and `dbus` concepts.

Important Files, Terms, and Utilities

- **/sys/**
- **/proc/**
- **/dev/**
- `modprobe`
- `lsmod`
- `lspci`
- `lsusb`

Lesson Overview

Modern computers rely on standards for firmware and hardware interaction. On x86 platforms, the firmware could be traditional **BIOS** or newer **UEFI**. Both allow for configuring hardware resources (e.g., integrated peripherals, IRQs, DMA settings) even before the operating system loads.

Once Linux is running, device detection and configuration rely on the kernel and support from user-space utilities such as `lspci`, `lsusb`, `lsmod`, and various pseudo-filesystems in /**proc** and /**sys**.

1. BIOS and UEFI Configuration

- **Accessing Firmware:** Typically press `Del`, `F2`, or `F12` at startup.
- **Common Configurations:**
  - Enable/disable integrated peripherals (USB ports, onboard audio, etc.).
  - Set boot order and define the primary device for the bootloader.
  - Adjust CPU features or RAM parameters if needed.
- **Impact:** Misconfiguration (e.g., wrong boot device) can prevent the OS from loading.

2. Device Detection in Linux

- **Goal:** Match hardware parts to the correct driver (**kernel module**).
- **Basic Workflow:**
  1. **Check if hardware is detected** (e.g., `lspci`, `lsusb`).
  2. **Verify if a driver is loaded** (e.g., `lsmod`, `lspci -k`).
  3. **Confirm functionality** via logs, testing, or additional tools.

3. Commands for Hardware Inspection

1. `lspci`
   - Lists PCI devices (graphics cards, network interfaces, etc.).
   - Use `-v` for more detail and `-k` to see which kernel modules are in use.
   - Example:

```
lspci -s 04:02.0 -v
lspci -s 01:00.0 -k
```

2. `lsusb`

- Lists USB devices (keyboards, mice, USB hubs, etc.).
- Use `-v` for verbose output and `-d <vendor:product>` to focus on a specific device.
- Example:

```
lsusb -v -d 1781:0c9f
lsusb -t # Show devices in a tree structure
```

3. `lsmod`
   - Shows loaded kernel modules.
   - Columns: **Module**, **Size**, **Used by** (dependency information).
   - Example:

```
lsmod | grep snd_hda_intel
```

4. `modprobe`
   - Loads or unloads modules (with dependencies).
   - `modprobe -r <module>` removes a module if not in use.
   - `modinfo <module>` shows module details (author, license, parameters, etc.).
   - Configuration files in `/etc/modprobe.d/` can blacklist or set module parameters.

4. Hardware Information Files

- **/proc** (pseudo-filesystem for processes and hardware info)
  - `/proc/cpuinfo`, `/proc/interrupts`, `/proc/ioports`, `/proc/dma`
- **/sys** (`sysfs` for device and kernel data)
- **/dev** (device files)
  - Each entry represents a device (e.g., `/dev/sda1`, `/dev/fd0`).
  - `udev` dynamically creates/removes these files as devices connect or disconnect.

5. Storage Devices

- **Block Devices:** Accessed in fixed-size blocks (hard disks, SSDs, etc.).
- **Naming Conventions:**
  - Newer kernels use `sd` prefix for most disks; partitions are numbered (`/dev/sda1`).

  - **IDE** devices also appear as `sd` on modern kernels
  - **NVMe** devices get names like `/dev/nvme0n1p1`.
  - **SD Cards** often appear as `/dev/mmcblk0p1`.

- **Hotplug and Coldplug:**
  - **Hotplug:** device recognized after boot (e.g., USB).
  - **Coldplug:** device recognized during boot (built-in or already connected).

Workbook Exercises

1. **Accessing BIOS/UEFI**
   - Reboot a test machine and enter BIOS/UEFI.
   - Locate the sections that let you enable/disable integrated peripherals.
   - Identify the menu where boot order is set.
2. **Listing Hardware**
   - On a Linux system, run `lspci -k`.
     - Identify which driver is used by the video card.
   - Run `lsusb -t`.
     - Check which USB driver modules are in use (e.g., `btusb, usbhid`).
3. **Exploring /proc and /sys**
   - View CPU details with `cat /proc/cpuinfo`.
   - Inspect interrupts with `cat /proc/interrupts`.
   - Explore `/sys/class` and `/sys/block` to see how devices are represented.
4. **Managing Kernel Modules**
   - Use `lsmod` to list all loaded modules.
   - Pick a module (e.g., a sound driver) and unload it with `sudo modprobe -r <module>`.
     - Check if removal is allowed (the module should not be in use).
   - Use `modinfo -p <module>` to see possible parameters, and note how you might apply them in `/etc/modprobe.d/`.
5. **Blacklisting a Module**
   - Create a test file in `/etc/modprobe.d/` to blacklist an unwanted module (e.g., `nouveau`).
   - Reboot and confirm it is not loaded by checking `lsmod`.

Summary

- Modern systems rely on firmware (BIOS/UEFI) for initial hardware configuration.
- Linux identifies devices via kernel modules; tools like `lspci`, `lsusb`, `lsmod`, and `modprobe` allow you to inspect and manage hardware.
- `/proc` and `/sys` provide detailed, real-time system information, while `udev` dynamically manages device nodes in `/dev`.
- Storage device naming conventions follow standard patterns such as `sd`, `nvme`, `mmcblk`, and partition numbers like `/dev/sda1`.
- Understanding how to enable/disable devices, load/unload modules, and explore hardware information files is crucial for effective system administration and LPIC-1 success.

Multiple-Choice Questions for 101.1

1. When trying to enable or disable motherboard-integrated peripherals, which component of the system is typically used?

   A) The BIOS or UEFI configuration utility

   B) The Linux kernel's initrd

   C) The `/boot` partition

   D) The `lsusb` command

2. Which command lists devices currently connected to the PCI bus?

   A) `modprobe`

   B) `lsmod`

   C) `lspci`

   D) `lshw`

3. Which of the following commands helps you list USB devices in a tree-like hierarchy?

   A) lsusb -a

   B) lsusb -s

   C) lsusb -f

   D) lsusb -t

4. To remove a kernel module (along with its dependencies) while the system is running, which command should be used?

A) modinfo -r

B) modprobe -r

C) rmmod –all

D) lsmod -r

5. On modern Linux systems, SATA disks are generally identified as which kind of device name?

   A) /dev/sdX

   B) /dev/hdX

   C) /dev/nvmeXnY

   D) /dev/fdX

6. Which file below would you edit to permanently blacklist a problematic kernel module such that it doesn't load automatically?

   A) /etc/rc.local

   B) /etc/modprobe.d/blacklist.conf

   C) /boot/grub/grub.cfg

   D) /proc/blacklist/modules

7. Which pseudo-filesystem is most specifically devoted to storing device and kernel data related to hardware?

   A) /dev

   B) /proc

   C) /sys

   D) /home

8. Which command line will show a specific USB device's verbose information using its vendor:product ID (e.g., 1781:0c9f)?

   A) lsusb -d 1781:0c9f -v

   B) lsusb -p 1781:0c9f -v

   C) lsusb -i 1781:0c9f

   D) lsusb -v -s 01:02

9. In the output of lsmod, the "Used by" column indicates:

   A) the file size of the module on disk

   B) the user-level applications that installed the module

C) the modules or processes depending on that module

D) kernel version compatibility for that module

10. If you need to confirm which kernel driver is in use by a particular PCI device, which `lspci` option combination is most helpful on recent distributions?

A) lspci -m

B) lspci -k

C) lspci -D

D) lspci –driver

11. What does the output of `lsusb -t` specifically highlight that differs from plain `lsusb`?

A) The exact partition layout of attached USB drives

B) A hierarchical (tree-like) representation of USB devices and drivers

C) The SCSI ID mappings of USB-attached devices

D) A summary of device's kernel modules only

12. Which best describes the function of the `modinfo` command?

A) It removes the specified module from the kernel

B) It displays all processes currently using a kernel module

C) It lists detailed information about a specified module, including parameters

D) It inserts the specified module and resolves dependencies

13. What is the role of `udev` on a modern Linux system?

A) It is a pseudo-filesystem used to track hardware devices in `/sys`

B) It permanently stores device drivers in `/boot`

C) It manages device nodes in `/dev`, handling hotplug/coldplug events

D) It only configures CPU frequency scaling

14. Which file inside `/proc` would you inspect to see how many interrupts have occurred for each device?

A) `/proc/ioports`

B) `/proc/dma`

C) `/proc/cpuinfo`

D) `/proc/interrupts`

15. If a device is recognized by the kernel but not functioning correctly, which of the following is the most likely underlying cause?

A) The BIOS is not set to read the device's firmware

B) The associated kernel module (driver) is not loaded or is misconfigured

C) The CPU lacks the required SSE instruction set

D) The device was not assigned a correct IRQ in the `/etc/fstab`

16. Which file is typically used to pass persistent module load options like `options nouveau modeset=0`?

A) `/etc/udev/rules.d/99-custom.rules`

B) `/proc/meminfo`

C) `/etc/modprobe.d/<module>.conf`

D) `/etc/modules-load.d/module.options`

17. What is the main purpose of SysFS (`/sys`) in a Linux system?

A) Stores process information like CPU usage

B) Holds user configuration data for `/home`

C) Exports device and driver information from the kernel to user space

D) Contains scripts to mount all system filesystems

18. Which command is most appropriate for listing all currently loaded kernel modules?

A) `ls -la /lib/modules/$(uname -r)`

B) `depmod -a`

C) `lsmod`

D) `insmod`

19. To selectively unload the `snd-hda-intel` module along with related dependent modules, which command would you use?

A) `modinfo snd-hda-intel -remove`

B) `lsmod -unload snd-hda-intel`

C) `depmod -r snd-hda-intel`

D) `modprobe -r snd-hda-intel`

20. If you see a disk labeled as `/dev/mmcblk0p1`, which type of physical device is this likely referring to?

A) A SATA SSD

B) An older IDE HDD

C) An SD card or MMC device

D) A USB DVD drive

Fill-in-the-Blank Questions for 101.1

1. The older firmware commonly used before the UEFI standard is called _____.
2. The _____ command lists all kernel modules currently loaded into the system.
3. A kernel module responsible for controlling hardware in Linux is often referred to as a _____.
4. The Linux subsystem that manages device node creation in `/dev` and handles hotplug/coldplug events is called _____.
5. The special, memory-based filesystem used for storing process and hardware information is the _____ directory.
6. To configure boot device priority and enable or disable onboard peripherals, a user must typically access the _____ or UEFI setup utility.
7. In Linux, disks commonly appear under `/dev` as _____ devices (e.g., `/dev/sda`, `/dev/sdb`) on modern systems.
8. The _____ command is used to insert or remove kernel modules and their dependencies.
9. When blacklisting a kernel module to prevent it from loading automatically, the configuration file is often placed in _____.
10. To see a hierarchical (tree-like) view of USB devices and the drivers handling them, you can run _____ with the `-t` option.

101.2 Boot the System

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 101.2**
- **Weight:** 3

Key Knowledge Areas

- Providing common bootloader commands and kernel options at boot.
- Understanding the boot sequence (BIOS/UEFI through OS startup).
- Familiarity with SysVinit, systemd, and Upstart.
- Checking boot events and logs (`dmesg`, `journalctl`).

Important Files, Terms, and Utilities

- **dmesg**
- **journalctl**
- **BIOS** / **UEFI**
- **bootloader** (GRUB)
- **kernel**
- **initramfs**
- **init** (SysVinit, systemd, Upstart)
- **/proc/cmdline**
- **/var/log/**

Lesson Overview

Booting a Linux system involves multiple stages:

1. **Firmware Load:** BIOS or UEFI initializes basic hardware.
2. **Bootloader:** Typically **GRUB**, which locates and loads the kernel.
3. **Kernel & initramfs:** Kernel initializes hardware and reads modules from the initramfs.
4. **System Initialization: init** (SysVinit, systemd, Upstart) starts services and completes the boot process.

1. BIOS vs. UEFI

- **BIOS**
  - Uses MBR (first 512 bytes) to load boot code (GRUB stage 1).
  - Relies on a DOS partition scheme and the Master Boot Record.
  - Boots the second stage of the bootloader, which in turn loads the kernel.
- **UEFI**
  - Looks at entries in **NVRAM** to find an **EFI application** (usually GRUB).
  - Loads the EFI application from a dedicated **EFI System Partition (ESP)**.
  - Supports **Secure Boot** to allow only signed EFI applications.

2. Bootloader (GRUB)

- Presents a menu of installed kernels or operating systems.
- Enables passing **kernel parameters** (e.g., `quiet`, `acpi=off`, `root=/dev/sdaX`, etc.).
- Kernel parameters can be made persistent in `/etc/default/grub` and then updated with:

```
grub-mkconfig -o /boot/grub/grub.cfg
```

- Current kernel parameters are visible in `/proc/cmdline`.

3. System Initialization

1. **initramfs**
   - Temporary root filesystem with essential drivers/modules.
   - Lets the kernel mount the actual root filesystem.
2. **init**
   - The "first process" in user space.
   - **SysVinit:** uses **runlevels** (0–6).
   - **systemd:** uses **targets**, concurrency, D-Bus, cgroups. Most common in modern distros.
   - **Upstart:** parallel boot focusing on faster startup. Largely replaced by systemd.

4. Boot Logging and Inspection

- **dmesg**
  - Displays the **kernel ring buffer** (including boot messages).
  - Clears with `dmesg -clear`.
- **journalctl**
  - Systemd-based logging tool.
  - `journalctl -b` shows current boot messages.
  - `journalctl -list-boots` lists previous boots.
- Traditional log files also found in `/var/log/`, e.g., `/var/log/messages` or `/var/log/syslog`.

Workbook Exercises

1. **Firmware Awareness**
   - Reboot a test machine.
   - Determine whether it uses **BIOS** or **UEFI**.
   - In BIOS: Find where the boot order is set.
   - In UEFI: Locate the ESP partition and explore contents if possible.
2. **GRUB Menu and Kernel Parameters**
   - Boot into the GRUB menu by pressing **Shift** (BIOS) or **Esc** (UEFI).
   - Edit a menu entry to add or change a kernel parameter (e.g., `init=/bin/bash`, `acpi=off`).
   - After boot, check `/proc/cmdline` to confirm your changes.
3. **System Initialization Tools**
   - Identify which init system your distribution uses (`ps -p 1 -o comm=`).
   - If it's systemd, compare output of these commands:

```
systemctl list-units --type=service
journalctl -b
```

   - If SysVinit is present, inspect runlevel scripts in `/etc/rc.d/` or `/etc/init.d/`.
4. **Inspecting Boot Logs**
   - Run `dmesg | less` to page through the kernel ring buffer.

- If using systemd, run `journalctl -list-boots` to see previous boots.
- View the logs for the current boot with `journalctl -b 0`.

5. **initramfs Exploration**
   - Locate your initramfs file (commonly in `/boot`, e.g., `initramfs-<version>.img`).
   - List contents using `lsinitrd` or `unmkinitramfs` (may require additional packages).
   - Identify which modules are included for the root filesystem.

Summary

- The boot process starts with **BIOS/UEFI** firmware, which calls **GRUB** to load the **kernel**.
- The **initramfs** contains essential modules and mounts the real root filesystem.
- An **init** system (SysVinit, systemd, Upstart) then starts daemons and services.
- **dmesg** and **journalctl** provide essential logs for troubleshooting.
- Understanding these steps ensures you can troubleshoot common startup issues and manage kernel parameters effectively.

101.3 Change Runlevels / Boot Targets and Shutdown or Reboot System

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 101.3**
- **Weight:** 3

Key Knowledge Areas

- Setting the default runlevel/boot target.
- Changing between runlevels/targets, including single-user mode.
- Shutting down and rebooting from the command line.
- Alerting users before switching runlevels/boot targets or major system events.
- Properly terminating processes.
- Awareness of **acpid** (power management).

Important Files, Terms, and Utilities

- **/etc/inittab** (SysVinit)
- **shutdown**
- **init**, **telinit** (SysVinit)
- **/etc/init.d/** (SysVinit scripts)
- **systemd**, **systemctl**
- **/etc/systemd/**, **/usr/lib/systemd/**
- **wall** (send messages to all logged-in users)

Lesson Overview

Linux can operate in different "states" or "modes" called **runlevels** in SysVinit or **targets** in systemd. Being able to switch between them and perform system shutdowns or reboots is essential for system administration.

1. SysVinit Runlevels

1. Runlevels

- **0** – Shutdown
- **1 (single), s** – Single-user (maintenance) mode
- **2, 3, 4** – Multi-user modes (3 is typical, 2/4 vary by distro)
- **5** – Multi-user plus graphical mode
- **6** – Reboot

2. Configuration

- **/etc/inittab** defines default runlevel (`id:x:initdefault:`)
- Each runlevel has a dedicated directory: **/etc/rc0.d/**, **/etc/rc1.d/**, etc.
- Scripts in **/etc/init.d/** are symlinked to these runlevel directories.
  - Names starting with **S** start services.
  - Names starting with **K** kill services.

3. Switching Runlevels

- **init** or **telinit** commands set the current runlevel.
- `telinit 1`: move to runlevel 1 (maintenance mode).
- `runlevel`: shows current and previous runlevel (e.g., `N 3` means currently 3 and no prior change).

4. Reloading /etc/inittab

- After editing **/etc/inittab**, run `telinit q` to re-read the config.

2. systemd Targets

1. systemd Concepts

- **Units** represent services, sockets, devices, mounts, automounts, targets, and snapshots.
- **systemctl** is the primary command to manage these units (start, stop, enable, etc.).

2. Targets

- systemd uses **targets** to group units. Examples:
  - **multi-user.target** – analogous to runlevel 3 (no GUI).
  - **graphical.target** – analogous to runlevel 5 (GUI mode).
- You can isolate a target:

```
systemctl isolate multi-user.target
```

3. Default Target

- Change default target:

```
systemctl set-default multi-user.target
```

- View current default:

```
systemctl get-default
```

- Avoid pointing to **shutdown.target** or **reboot.target**.

4. Service Management

- `systemctl start/stop/restart <service>.service`
- `systemctl enable/disable <service>.service` (at boot)
- `systemctl status <service>.service`
- `systemctl list-unit-files -type=service` – list available services
- `systemctl list-units -type=service` – list loaded/running services

5. Power Management

- `systemctl suspend`, `systemctl hibernate`
- For finer power-event control (e.g., lid close), **acpid** can be used instead of systemd's built-in power management.

3. Upstart (Historical)

1. **Upstart** was used in older Ubuntu-based systems before switching to systemd.
2. **Commands**:
   - `initctl list` – list services and states
   - `start / stop / status <service>` – control services
   - Initialization scripts: **/etc/init/**
3. `runlevel` and `telinit` still work for basic runlevel tasks.

4. Shutting Down and Rebooting

1. shutdown

- Syntax:

```
shutdown [option] time [message]
```

- **time** can be `now`, `+m` (minutes from now), or `hh:mm` (absolute time).
- Common options:
  - **-h** – halt/power off
  - **-r** – reboot
- Notifies logged-in users and prevents new logins (unless overridden).

2. systemctl (systemd)

- `systemctl reboot` – reboot system
- `systemctl poweroff` – power off system
- Sometimes distros alias `poweroff` and `reboot` to systemd commands.

3. wall

- Sends a message to all logged-in users' terminals (similar to `shutdown`'s broadcast).
- Useful for manual warnings before switching to single-user mode or shutting down.

Workbook Exercises

1. **Identify Your Init System**
   - Run `ps -p 1 -o comm=` to see if your system uses **systemd**, **init**, or **Upstart**.
2. **Practice Switching Runlevels (SysV)**
   - On a SysVinit system, edit **/etc/inittab** to set default runlevel to **3**.
   - Run `telinit q` and verify with `runlevel`.
   - Switch to single-user mode: `telinit 1`.
3. **Practice Managing systemd Targets**
   - Show the current default target: `systemctl get-default`.
   - Switch from **graphical.target** to **multi-user.target** using:

```
systemctl isolate multi-user.target
```

   - Confirm the change: `systemctl status multi-user.target`.
4. **Service Control with systemd**
   - Start a service (e.g., `ssh.service`):

```
sudo systemctl start ssh
```

   - Check service status: `systemctl status ssh`.
   - Enable service at boot: `systemctl enable ssh`.
5. **Shutdown Commands**
   - Schedule a reboot in 10 minutes, sending a warning message:

```
sudo shutdown -r +10 "System will reboot in 10 minutes."
```

   - Cancel a scheduled shutdown with:

```
sudo shutdown -c
```

   - Use **systemctl** to reboot immediately: `systemctl reboot`.
6. **Sending Warnings**
   - Open a second terminal and log in as a test user.

- From the admin terminal, run:

```
wall "Warning! System moving to single-user mode in 1 minute."
```

- Confirm the message appears in the other terminal.

Summary

- **SysVinit** uses numbered runlevels (0–6), configured via **/etc/inittab**, and manages services in **/etc/init.d/**.
- **systemd** uses **targets** and **units**, with **systemctl** providing service control and target isolation.
- **Upstart** (historical) uses **initctl** and scripts in **/etc/init/**.
- Shutting down, rebooting, or switching modes should alert current users (via **wall** or **shutdown**'s broadcast).
- Proper runlevel/target configuration ensures the correct set of services starts at boot, maximizing system stability and user support.

Chapter 2

Topic 102: Linux Installation and Package Management

2.1 102.1 Design hard disk layout

[Brief syllabus and questions to be added here]

2.2 102.2 Install a boot manager

[Brief syllabus and questions to be added here]

2.3 102.3 Manage shared libraries

[Brief syllabus and questions to be added here]

2.4 102.4 Use Debian package management

[Brief syllabus and questions to be added here]

2.5 102.5 Use RPM and YUM package management

[Brief syllabus and questions to be added here]

2.6 102.6 Linux as a virtualization guest

[Brief syllabus and questions to be added here]

Chapter 3

Topic 103: GNU and UNIX Commands

3.1 103.1 Work on the command line

[Brief syllabus and questions to be added here]

3.2 103.2 Process text streams using filters

[Brief syllabus and questions to be added here]

3.3 103.3 Perform basic file management

[Brief syllabus and questions to be added here]

3.4 103.4 Use streams, pipes and redirects

[Brief syllabus and questions to be added here]

103.4 Lesson 1

[Brief syllabus and questions to be added here]

3.5 103.5 Create, monitor and kill processes

[Brief syllabus and questions to be added here]

3.6 103.6 Modify process execution priorities

[Brief syllabus and questions to be added here]

103.6 Lesson 1

[Brief syllabus and questions to be added here]

3.7 103.7 Search text files using regular expressions

[Brief syllabus and questions to be added here]

3.8 103.8 Basic file editing

[Brief syllabus and questions to be added here]

Chapter 4

Topic 104: Devices, Linux Filesystems, Filesystem Hierarchy Standard

4.1 104.1 Create partitions and filesystems

[Brief syllabus and questions to be added here]

4.2 104.2 Maintain the integrity of filesystems

[Brief syllabus and questions to be added here]

4.3 104.3 Control mounting and unmounting of filesystems

[Brief syllabus and questions to be added here]

4.4 104.5 Manage file permissions and ownership

[Brief syllabus and questions to be added here]

104.5 Lesson 1

[Brief syllabus and questions to be added here]

4.5 104.6 Create and change hard and symbolic links

[Brief syllabus and questions to be added here]

4.6 104.7 Find system files and place files in the correct location

[Brief syllabus and questions to be added here]

Chapter 5

Topic 105: Shells and Shell Scripting

5.1 105.1 Customize and use the shell environment

[Brief syllabus and questions to be added here]

5.2 105.2 Customize or write simple scripts

[Brief syllabus and questions to be added here]

Chapter 6

Topic 106: User Interfaces and Desktops

6.1 106.1 Install and configure X11

[Brief syllabus and questions to be added here]

6.2 106.2 Graphical Desktops

[Brief syllabus and questions to be added here]

6.3 106.3 Accessibility

[Brief syllabus and questions to be added here]

Chapter 7

Topic 107: Administrative Tasks

7.1 107.1 Manage user and group accounts and related system files

[Brief syllabus and questions to be added here]

7.2 107.2 Automate system administration tasks by scheduling jobs

[Brief syllabus and questions to be added here]

7.3 107.3 Localisation and internationalisation

[Brief syllabus and questions to be added here]

Chapter 8

Topic 108: Essential System Services

8.1 108.1 Maintain system time

[Brief syllabus and questions to be added here]

8.2 108.2 System logging

[Brief syllabus and questions to be added here]

8.3 108.3 Mail Transfer Agent (MTA) basics

[Brief syllabus and questions to be added here]

8.4 108.4 Manage printers and printing

[Brief syllabus and questions to be added here]

Chapter 9

Topic 109: Networking Fundamentals

9.1 109.1 Fundamentals of internet protocols

[Brief syllabus and questions to be added here]

9.2 109.2 Persistent network configuration

[Brief syllabus and questions to be added here]

9.3 109.3 Basic network troubleshooting

[Brief syllabus and questions to be added here]

9.4 109.4 Configure client side DNS

[Brief syllabus and questions to be added here]

Chapter 10

Topic 110: Security

10.1 110.1 Perform security administration tasks

[Brief syllabus and questions to be added here]

10.2 110.2 Setup host security

[Brief syllabus and questions to be added here]

10.3 110.3 Securing data with encryption

[Brief syllabus and questions to be added here]

Answers

Topic 101: System Architecture

101.1 Determine and Configure Hardware Settings

Multiple-Choice Questions (101.1)

1. A
2. C
3. D
4. B
5. A
6. B
7. C
8. A
9. C
10. B
11. B
12. C
13. C
14. D
15. B
16. C
17. C
18. C

19. D
20. C

Fill-in-the-Blank Questions (101.1)

1. BIOS
2. lsmod
3. driver
4. udev
5. /proc
6. BIOS
7. SCSI
8. modprobe
9. /etc/modprobe.d
10. lsusb

101.2 Boot the System

Multiple-Choice Questions (101.2)

1. **D**
2. **A**
3. **C**
4. **B**
5. **A**

Fill-in-the-Blank Questions (101.2)

1. **kernel**
2. **grub**
3. **/boot**
4. **initrd**
5. **systemd**