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.

Multiple-Choice Questions for 101.2

1. Which of the following best describes the role of the **kernel ring buffer** during the boot process?

   A) It stores a copy of the MBR after BIOS initialization.

   B) It holds user processes' initialization scripts during startup.

   C) It temporarily stores kernel messages, including boot messages.

   D) It provides secure boot verification for the EFI System Partition.

2. On a typical Linux system with GRUB, which file should be edited to **persistently** add kernel boot parameters?

   A) `/etc/default/grub`

   B) `/etc/systemd/system.conf`

   C) `/boot/vmlinuz`

   D) `/proc/cmdline`

3. Which bootloader is most commonly associated with modern x86-based Linux systems?

   A) LILO

   B) SYSLINUX

   C) BURG

   D) GRUB

4. Which of the following statements about **Secure Boot** is **true**?

   A) It forces the user to boot only from a local disk rather than USB devices.

   B) It requires EFI applications to be signed/authorized by the hardware vendor or a trusted party.

   C) It loads the SysVinit scripts in parallel to reduce the boot time of the OS.

   D) It uses MBR partition tables exclusively and disables GPT.

5. The BIOS in a legacy (non-UEFI) x86 system typically reads and executes boot code from what specific location?

   A) The first 440 bytes of the MBR on the primary boot device

   B) The second stage of GRUB in `/boot/grub`

   C) The NVRAM partition labeled `/efi/boot`

   D) `/boot` partition

6. What is the **primary purpose** of `initramfs` during the boot process?

   A) To store the kernel ring buffer.

   B) To provide early user accounts for system security.

   C) To load required kernel modules so the real root filesystem can be mounted.

   D) To replace the BIOS firmware in older systems.

7. You want to limit a Linux guest system to a maximum of 1 GB of RAM at boot time. Which kernel parameter should be used?

   A) `nosmp=1G`

   B) `mem=1G`

   C) `ram=1G`

   D) `maxcpus=1G`

8. Which of the following is a feature of **systemd**?

   A) Entirely depends on runlevels 0–6 and SysV scripts.

   B) Uses sockets and D-Bus for on-demand service activation.

   C) Must be installed as a kernel module.

   D) It can only run one service at a time to avoid concurrency issues.

9. While troubleshooting a boot issue, you want to see **previous** system boots' log messages. Which systemd-related command enables you to do this?

   A) `dmesg -previous`

   B) `journalctl -list-boots`

   C) `systemctl -history`

   D) `logrotate -b`

10. After you edit `/etc/default/grub` to add a new kernel parameter, which command is typically used to **update** the GRUB configuration on many distributions?

    A) `cp /etc/default/grub /boot/grub/grub.conf`

    B) `touch /boot/grub/grub.cfg`

    C) `grub-install /boot`

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

11. What does the kernel parameter `acpi=off` do?

    A) Disables multi-processor support, similar to `nosmp`.

    B) Disables BIOS POST checks and loads the kernel directly.

C) Disables ACPI functions to troubleshoot power management or ACPI-related issues.

D) Forces the root filesystem to be mounted as read-only.

12. In a SysVinit-based system, which file primarily determines which **runlevel** the system will go to when it finishes booting?

A) `/etc/fstab`

B) `/boot/initramfs-<version>.img`

C) `/etc/inittab`

D) `/var/log/boot.log`

13. When using UEFI, which partition **must** contain the bootloader or EFI applications?

A) The root (`/`) filesystem partition

B) A dedicated GPT partition labeled "MBR"

C) An NVRAM-based partition called `/var/lib/EFI`

D) The EFI System Partition (ESP)

14. Which kernel parameter instructs the system to **start** a different **initial process** instead of the default `/sbin/init` or systemd?

A) `init=/bin/bash`

B) `systemd.unit=multi-user.target`

C) `noapic`

D) `ro`

15. The term **daemon** is typically used to describe which kind of program in a Linux system?

A) A program that only runs once at boot and then terminates.

B) A service that remains **running** in the background.

C) Any script that an administrator invokes manually from the command line.

D) A background service process (e.g. system or network) that runs indefinitely.

16. Which of the following is **not** a valid kernel parameter for controlling the amount of displayed boot information?

A) `verbose=0`

B) `quiet`

C) `vga=ask`

D) `maxcpus=1`

17. If a critical system service fails to start during boot and the system uses **systemd**, where would you most likely check **first** for the relevant error messages?

    A) `/proc/cmdline`

    B) `/etc/default/grub`

    C) `systemctl list-jobs`

    D) `journalctl -b` or `journalctl -boot`

18. In a system that uses SysVinit, which runlevel is **commonly** used for **single-user mode** (maintenance mode)?

    A) 2

    B) 5

    C) 1

    D) 3

19. Which of the following statements about **Upstart** is correct?

    A) It can parallelize the initialization of services but has largely been replaced by systemd.

    B) It replaces the BIOS in older systems.

    C) It is strictly a tool for reading the kernel ring buffer.

    D) It is used to sign EFI applications for Secure Boot.

20. The BIOS POST (Power-On Self-Test) primarily checks for:

    A) Valid ext4 partitions on the system's boot drive.

    B) Basic hardware components and any major hardware failures.

    C) Corrupted kernel parameters in `/proc/cmdline`.

    D) Upstart jobs that should be started first.

Fill-in-the-Blank Questions for 101.2

1. The firmware on modern x86 systems can be either traditional _____ or the more advanced _____.

2. On legacy BIOS-based systems, the first stage of the bootloader is typically located in the first _____ bytes of the _____.

3. When using UEFI, the bootloader or EFI applications are stored in a dedicated partition called the _____, often formatted with a FAT filesystem.

4. The kernel parameter _____`=/bin/bash` causes the system to start a Bash shell as the first user-space process instead of the standard init system.

5. The file `/etc/default/grub` contains the directive `GRUB_CMDLINE_LINUX`, which is used to specify _____ passed to the kernel at boot time.

6. The command `grub-mkconfig -o /boot/grub/grub.cfg` is needed after modifying `/etc/default/grub` to _____ the bootloader configuration.

7. The memory area that stores kernel messages, including boot information, is called the _____, which can be viewed with the `dmesg` command.

8. The _____ process runs basic hardware checks (like checking memory) as soon as the machine is powered on, before loading the bootloader.

9. In a SysVinit-based system, the file `/etc/`_____ typically defines which runlevel the system will enter when it finishes booting.

10. A(n) _____ is a background service or process that remains running to provide system or network functionality.

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.

Multiple-Choice Questions for 101.3

1. Which file traditionally defines the default runlevel in a SysVinit system?

   A) `/etc/inittab`

   B) `/etc/rc.conf`

   C) `/etc/systemd/system.conf`

   D) `/etc/default/runlevel`

2. In SysVinit, which runlevel usually corresponds to **system restart**?

   A) Runlevel 1

   B) Runlevel 3

   C) Runlevel 5

   D) Runlevel 6

3. Which command is used on a SysVinit system to **check the current runlevel**?

   A) `who -r`

   B) `runlevel`

   C) `init`

   D) `sysvcheck`

4. On a SysVinit system, which **runlevel** is typically reserved for **multi-user mode without a graphical environment**?

   A) Runlevel 0

   B) Runlevel 1

   C) Runlevel 3

   D) Runlevel 6

5. Which command **reloads** the `/etc/inittab` file after changes are made (on a SysVinit system)?

   A) `telinit q`

   B) `init reload`

   C) `systemctl daemon-reload`

   D) `reload runlevel`

6. Which **systemd unit type** is used for grouping other units so they can be controlled as a single entity?

   A) service

B) automount

C) target

D) socket

7. On a **systemd** system, which command would you use to **switch** the system to `multi-user.target` immediately?

   A) `systemctl default multi-user.target`

   B) `systemctl multi-user.target`

   C) `systemctl reload multi-user.target`

   D) `systemctl isolate multi-user.target`

8. Which command is commonly used on SysVinit systems to **change** the current runlevel **without** rebooting?

   A) `systemctl isolate`

   B) `telinit`

   C) `initctrl`

   D) `switchrun`

9. In a SysVinit layout, scripts in directories like `/etc/rc3.d/` typically **start** with what letter if they are launched upon entering that runlevel?

   A) R

   B) G

   C) S

   D) T

10. Which **runlevel** or mode is typically used for **maintenance** when the system is only available to the administrator (no network services)?

    A) Single-user (Runlevel 1)

    B) Graphical mode (Runlevel 5)

    C) Multi-user mode (Runlevel 3)

    D) Runlevel 2

11. Which **SysVinit** command can be used to **halt** the system, after modifying the `/etc/inittab` entry for Ctrl+Alt+Del with the `-a` option?

    A) `halt -a`

    B) `shutdown`

    C) `poweroff`

D) `stop system`

12. Which **systemctl** command would you use to **turn off** the system immediately on a **systemd** host?

    A) `systemctl shutdown`

    B) `systemctl down`

    C) `systemctl isolate runlevel0.target`

    D) `systemctl poweroff`

13. Which **systemd** unit type is used for hardware devices identified by the kernel?

    A) target

    B) service

    C) device

    D) mount

14. Which file is **not** used by **systemd** to set the default system target?

    A) `/etc/systemd/system/default.target`

    B) `/lib/systemd/system/multi-user.target`

    C) `/lib/systemd/system/graphical.target`

    D) `/etc/inittab`

15. If you see the output `tty5 start/running, process 1764` on an Ubuntu system, which **init system** is likely in use?

    A) SysVinit

    B) Upstart

    C) systemd

    D) OpenRC

16. On a **systemd** system, which command **reboots** the machine?

    A) `systemctl shutdown -r`

    B) `systemctl kill`

    C) `systemctl isolate reboot.target`

    D) `systemctl reboot`

17. Which **systemd** unit type is used to define an on-demand mount point?

    A) device

B) service

C) socket

D) automount

18. Which **Upstart** command is used to **stop** a currently running job or service?

    A) `upstartctl kill`

    B) `stop`

    C) `service halt`

    D) `haltjob`

19. Which command is typically used to **send a message** to all logged-in users' terminals?

    A) `wall`

    B) `announce`

    C) `globalmsg`

    D) `bcast`

20. In the **SysVinit** scheme, which directory contains startup scripts (symbolic links) specifically for **runlevel 2**?

    A) `/etc/init.d2/`

    B) `/etc/rc.d/2/`

    C) `/etc/rc2.d/`

    D) `/etc/sysvinit/2/`

Fill-in-the-Blank Questions for 101.3

1. In a **SysVinit** system, the default runlevel is configured in the file _____.

2. To switch the system to **single-user mode** (runlevel 1) on a SysVinit system, you can type _____ **1** or _____ **s**.

3. The command _____ **q** is used to make **init** re-read the **/etc/inittab** file after changes are made.

4. In **System V** style initialization, scripts controlling services are located in _____, while each runlevel (e.g., runlevel 3, 5) has its own subdirectory like **/etc/rc3.d/** or **/etc/rc5.d/**.

5. Under **systemd**, each background process or subsystem is referred to as a _____ (e.g., **httpd.service**).

6. To change the **default target** in **systemd** without editing kernel parameters directly, you can use the command **systemctl set-default** _____**.target**.

7. In **systemd**, if you want to switch to **multi-user mode** without rebooting, you can execute **systemctl** _____ **multi-user.target**.

8. When switching from **Upstart**, Ubuntu replaced its init system with _____.

9. The _____ command sends a message to the terminal sessions of all logged-in users and is useful before shutting down or switching runlevels.

10. In a **SysVinit** system, **Runlevel 0** corresponds to _____, while **Runlevel 6** corresponds to a **restart** of the system.

Chapter 2

Topic 102: Linux Installation and Package Management

102.1 Design Hard Disk Layout

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 102, Objective 102.1**
- **Weight:** 2

Key Knowledge Areas

- Allocating filesystems and swap space to separate partitions or disks.
- Tailoring the partitioning scheme to system usage.
- Understanding `/boot` or EFI System Partition requirements for booting.
- Basic familiarity with LVM (Logical Volume Manager).

Important Terms and Utilities

- `/` `(root)`, `/boot`, `/home`, `/var`
- **EFI System Partition (ESP)**
- **swap**
- **mount points** (e.g., `/mnt`, `/media/USER/LABEL`)
- **partitions** and **logical volumes**
- **LVM** (Physical Volumes, Volume Groups, Logical Volumes)

Lesson Overview

Designing an effective disk layout is critical for system stability, performance, and ease of administration. You must understand partitions, filesystems, mount points, swap, and how LVM can simplify storage allocation.

1. Partitions, Filesystems, and Mount Points

1. **Partitions**
    - Logical "fences" on a disk; each partition has its own filesystem.
    - Partition information is stored in the **partition table**.
    - Partitions **cannot** span multiple disks (unless using LVM or RAID).
2. **Filesystems**

- Define how data is organized in directories, files, and metadata.
- Must be **mounted** on a mount point (e.g., `/mnt/mydata`).

3. **Mount Points**
   - Directory where a filesystem is attached.
   - Common directories:
     - `/mnt/` – traditional manual mount point.
     - `/media/` – automatic mounting of removable media.
   - Existing contents of a mount point become hidden while another filesystem is mounted.

2. Recommended Partitions and Their Uses

1. **Root Partition (/)**
   - Base of the Linux directory structure.
   - Typically holds OS binaries and system config if not separated elsewhere.
2. **`/boot` or EFI System Partition (ESP)**
   - `/boot` stores bootloader files (kernel images, initramfs, GRUB).
   - ESP is used on UEFI systems (formatted as FAT).
   - Usually 200–300 MB in size is sufficient for either.
   - Keeping boot files separate can help ensure the system can still boot if root is damaged.
3. `/home`
   - Houses users' personal files and preferences.
   - Separating `/home` allows OS reinstallation without erasing user data.
   - Size depends on user data and expected usage.
4. `/var`
   - Contains variable data: logs (`/var/log`), caches (`/var/cache`), temp data (`/var/tmp`), etc.
   - On servers, `/var` can grow significantly (e.g., web or database data).
   - Putting `/var` on a separate partition (or disk) improves stability and prevents root from filling up.
5. **Swap**

- Extension of RAM to disk; cannot be mounted as a normal directory.
- Often sized according to usage (e.g., old rule was 2×RAM; modern guidelines vary).
- Consider **hibernation** requirements (swap ≥ RAM if hibernation is used).

3. LVM (Logical Volume Manager)

1. **Overview**
   - Provides flexible "virtual" partitions called **Logical Volumes (LVs)**.
   - **Physical Volumes (PVs)** → grouped into **Volume Groups (VGs)** → split into **Logical Volumes (LVs)**.
   - LVM allows resizing or adding storage more easily than traditional partitions.
2. **Advantages**
   - **Ease of extension:** add space without reformatting or migrating data.
   - **Abstracts** underlying physical disks.
   - Logical volumes appear in `/dev/VG_NAME/LV_NAME`.
3. **Basic Workflow (High-level)**
   (a) Create or identify a **partition** (or entire disk) as a PV (`pvcreate /dev/sdaX`).
   (b) Combine PVs into a **Volume Group** (`vgcreate MYVG /dev/sdaX`).
   (c) Create a **Logical Volume** (`lvcreate -L 20G -n MYSERVICELV MYVG`).
   (d) Format LV with a filesystem (`mkfs.ext4 /dev/MYVG/MYSERVICELV`).
   (e) Mount where desired (`/etc/fstab` entry or `mount` command).

Workbook Exercises

1. **Plan a Basic Partition Scheme**
   - Imagine you have a 500 GB disk for a personal workstation.
   - Sketch out your proposed partition table: `/boot` (300 MB), root (`/`), `/home`, `/var`, and swap.
   - Consider sizes for each partition and justify your choices.
2. **Identify ESP/BIOS Partitions**
   - On a UEFI-based system, locate and identify the **EFI System Partition** (`/boot/efi`).

- Check partition type using `gdisk -l /dev/sda` or `fdisk -l /dev/sda`.
- Verify its filesystem (FAT-based) with `lsblk -f` or `blkid`.

3. **Decide on Swap Size**
   - If your system has 8 GB of RAM, use Red Hat's guidelines to propose a recommended swap size.
   - If planning hibernation, recalculate.
4. **Mount Points**
   - Create a directory `/mnt/testmount`.
   - Using an existing spare partition (or loopback device), manually mount it on `/mnt/testmount`.
   - Verify it is mounted with `mount | grep /mnt/testmount`.
5. **LVM Planning**
   - Using a virtual environment with two disks, plan an LVM layout:
     (a) Convert one partition from each disk into PVs.
     (b) Create a Volume Group that spans both.
     (c) Create one or more Logical Volumes for `/data`.
   - Write down how you will format and mount `/data`.
6. **Storage Scenarios**
   - You run out of disk space on `/home`. What steps can you take with LVM to add more space?
   - If `/var` was not separated and you frequently run out of space due to logs, how might you redesign?

Summary

- **Partitions** define logical divisions of a disk, while **filesystems** define how data is stored.
- Strategic partitioning improves **stability, security, and maintenance** (e.g., `/boot`, `/home`, `/var` separate).
- **UEFI** systems require an **EFI System Partition (ESP)**; BIOS systems may benefit from a separate `/boot`.
- Adequate **swap** is essential; guidelines depend on RAM, system usage, and whether hibernation is used.
- **LVM** adds flexibility for resizing and pooling storage among multiple physical disks.

102.2 Install a Boot Manager

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 102, Objective 102.2**
- **Weight:** 2

Key Knowledge Areas

- Providing alternate or backup boot options.
- Installing and configuring boot loaders (GRUB Legacy, GRUB 2).
- Performing basic GRUB 2 configuration changes.
- Interacting with the boot loader at startup.

Important Files, Terms, and Utilities

- **MBR** (Master Boot Record)
- `/boot` directory or partition (often containing GRUB files, kernels, initrd)
- `menu.lst`, `grub.cfg`, and `grub.conf`
- `grub-install`, `grub-mkconfig` (or `update-grub`)
- **chainloading** (for non-Linux OS, e.g., Windows)

Lesson Overview

A system's boot loader is the first software executed when a machine powers on. On Linux, this is typically **GRUB** (either Legacy or GRUB 2). GRUB loads the kernel and passes control to it. Having a working knowledge of installing and configuring GRUB is essential for system recovery and customizing boot behavior.

1. GRUB Legacy vs. GRUB 2

1. **GRUB Legacy**
    - Older, no longer actively developed (last release 0.97 from 2005).
    - Configuration file: `/boot/grub/menu.lst` (sometimes `grub.conf`).
    - Simpler configuration, fewer features.
2. **GRUB 2**

- Complete rewrite, default on most modern distributions.
- Configuration files:
  - `/etc/default/grub` (main user-editable file)
  - `/boot/grub/grub.cfg` (auto-generated, do not edit manually)
- More modular, supports more filesystems, better scripting, theming, etc.

2. Bootloader Locations and Partitions

1. **MBR Partition Scheme**
   - Legacy layout where the first 512 bytes of the disk contain the MBR (boot code + partition table).
   - Boot loader code often placed in MBR + post-MBR gap (32 KB) before the first partition.
2. **GPT (GUID Partition Table)**
   - Modern layout for large disks (>2 TB).
   - Requires a **BIOS boot partition** (for BIOS systems) or uses **EFI System Partition (ESP)** on UEFI systems.
3. **/boot Partition**
   - Often first partition on the disk, historically to avoid BIOS cylinder limits.
   - Typically ∼300 MB in size, containing kernel images (`vmlinuz`), initrd, GRUB files, etc.
   - Helps ensure boot files remain accessible (especially if `/` uses encryption or an unsupported filesystem).

3. Installing GRUB 2

1. `grub-install`
   - Installs GRUB 2 boot code onto a disk (e.g., `/dev/sda`) or EFI partition.
   - Syntax examples:

```
grub-install --boot-directory=/boot /dev/sda
# or for a dedicated /boot partition mounted at /mnt/tmp:
grub-install --boot-directory=/mnt/tmp /dev/sda
```

   - Must point to the **disk** (e.g., `/dev/sda`), not a specific partition (unless UEFI requires otherwise).

2. **Configuration**

   - `/etc/default/grub` – main file for user edits. Common parameters:
     - `GRUB_DEFAULT`: default menu entry (0-based index, or `saved`).
     - `GRUB_SAVEDEFAULT`: if set to `true` with `GRUB_DEFAULT=saved`, boots the last-chosen entry.
     - `GRUB_TIMEOUT`: seconds before auto-booting the default. `-1` waits indefinitely.
     - `GRUB_CMDLINE_LINUX`: universal kernel parameters (e.g., `quiet`, `splash`).
   - `grub-mkconfig` (or `update-grub`) generates `/boot/grub/grub.cfg` from the above file and scripts in `/etc/grub.d/`:

```
grub-mkconfig -o /boot/grub/grub.cfg
# or:
update-grub
```

3. **Menu Entries**

   - Auto-discovered for Linux, other OS, or kernels.
   - Custom entries often added to `/etc/grub.d/40_custom`, then re-run `update-grub`.

4. **Interacting with GRUB 2**

   - **Boot Menu:** highlight an entry with arrow keys, press **e** to edit before booting.
   - **Shell Mode:** press **c** to access `grub>` shell.
   - **Rescue Shell** (`grub rescue>`): minimal commands, must `insmod` needed modules (e.g., `normal`, `linux`) if GRUB config is broken.

4. GRUB Legacy (for Reference)

1. **Installing**

   - Via `grub-install /dev/sda` (must specify the disk, not a partition).
   - From GRUB Legacy shell:

```
grub> root (hd0,0)
grub> setup (hd0)
```

   - `root (hd0,0)` means the first disk (`hd0`), first partition (`0`), if `/boot` is there.

2. **Configuration:** `/boot/grub/menu.lst`

   - Example menu entry:

```
title My Linux
root (hd0,0)
kernel /vmlinuz root=/dev/hda1
initrd /initrd.img
```

- `chainloader +1` used to boot Windows or other OS by loading their own bootloader code.

5. Booting from the GRUB Shell

1. **Identify Partitions**:

```
grub> ls
(hd0) (hd0,msdos1)
```

2. **Set root** (example):

```
grub> set root=(hd0,msdos1)
```

3. **Load Kernel & Initrd** (GRUB 2 example):

```
grub> linux /vmlinuz root=/dev/sda1
grub> initrd /initrd.img
grub> boot
```

4. **Rescue Mode**: need to `set prefix=(hd0,msdos1)/boot/grub` and `insmod normal`, `insmod linux` before proceeding.

Workbook Exercises

1. **Identify Boot Device**
   - Run `fdisk -l /dev/sda` or `lsblk -f` and find your **boot partition**.
   - Note which partition is marked as bootable.
2. **Install GRUB 2**
   - Mount your `/boot` (or boot partition) if needed at `/mnt/tmp`.
   - Run:

```
grub-install --boot-directory=/mnt/tmp /dev/sda
```

   - Verify GRUB files are placed in `/mnt/tmp/boot/grub`.
3. **Customize GRUB Timeout**

- Edit `/etc/default/grub` and set `GRUB_TIMEOUT=5`.
- Run `update-grub` (or `grub-mkconfig -o /boot/grub/grub.cfg`).
- Reboot and confirm you see the menu for 5 seconds.

4. **Add a Kernel Parameter**
   - In `/etc/default/grub`, add an option to `GRUB_CMDLINE_LINUX="quiet splash"`.
   - Update GRUB and reboot. Check `/proc/cmdline` to confirm the new parameter.
5. **Practice Chainloading**
   - If you have a Windows install at `(hd0,2)`, add a custom entry in `/etc/grub.d/40_custom` (or in GRUB Legacy's `menu.lst`):

```
menuentry "Windows" {
    set root=(hd0,2)
    chainloader +1
}
```

   - Update GRUB and verify you can boot into Windows.
6. **GRUB Rescue Simulation**
   - Temporarily rename `/boot/grub/grub.cfg` to break GRUB.
   - Reboot to force the `grub rescue>` prompt.
   - Use `ls`, `set prefix=`, `insmod normal`, etc., to recover manually.
   - Restore `grub.cfg` after testing.

Summary

- **GRUB 2** is the modern bootloader on most Linux systems, replacing **GRUB Legacy**.
- `grub-install` places boot code on the MBR (BIOS) or ESP (UEFI).
- `/etc/default/grub` and scripts in `/etc/grub.d/` define the GRUB 2 menu.
- Use `update-grub` (or `grub-mkconfig`) to regenerate `/boot/grub/grub.cfg`.
- In emergencies, the **GRUB shell** (normal or rescue) can manually load kernel and initrd to boot.

102.3 Manage Shared Libraries

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 102.3**
- **Weight:** 1

Key Knowledge Areas

- Identifying shared libraries.
- Understanding typical locations of system libraries.
- Loading and configuring shared libraries at runtime.

Important Commands and Files

- `ldd` – shows shared library dependencies.
- `ldconfig` – updates library cache and symbolic links.
- `/etc/ld.so.conf` and `/etc/ld.so.conf.d/` – configuration for library paths.
- `LD_LIBRARY_PATH` – environment variable to temporarily add library paths.

Lesson Overview

Shared libraries (`.so` files) allow multiple executables to reuse common code, reducing memory usage and disk size. Administrators must know how to locate libraries, configure library paths, and troubleshoot missing dependencies.

1. Concept of Shared Libraries

1. **Static Libraries (`.a`)**
    - Code is **copied** into an executable at compile/link time.
    - Larger file size; no external dependencies at runtime.
2. **Dynamic (Shared) Libraries (`.so`)**
    - Code is **not** copied into the executable.
    - Must be present at runtime.
    - More efficient memory usage (shared among processes).

2. Typical Library Naming and Locations

1. **Shared Library Naming**
   - Usually `libXYZ.so.major.minor`.
   - Example: `libc.so.6 → libc-2.24.so`.
   - Symbolic links often point from a generic name to a versioned file.
2. **Locations**
   - `/lib`, `/lib64`, `/usr/lib`, `/usr/local/lib`, and architecture-specific directories like `/lib/x86_64-linux-gnu`.
3. **Dynamic Linker**
   - `ld.so` or `ld-linux.so` handles runtime loading of `.so` files.

3. Configuring Library Paths

1. `/etc/ld.so.conf` **and** `/etc/ld.so.conf.d/*.conf`
   - Lists directories to be searched by the dynamic linker.
   - Usually references sub-files in `/etc/ld.so.conf.d/`.
2. `ldconfig`
   - Reads config files, creates symbolic links, updates `/etc/ld.so.cache`.
   - Run after installing new libraries or editing config.
   - Common options:
     - **-v**: verbose mode.
     - **-p**: print current cache contents.
3. `LD_LIBRARY_PATH`
   - Environment variable to **temporarily** add library directories.
   - Example:

```
export LD_LIBRARY_PATH=/usr/local/mylib
```

   - Similar to `PATH`, but for shared libraries.

4. Checking Dependencies with `ldd`

1. `ldd /path/to/executable`
    - Shows which `.so` files an executable needs and where they're loaded from.
    - Example:

```
ldd /usr/bin/git
```

2. `ldd /path/to/library.so`
    - Also works on `.so` files themselves.
3. **`-u` (unused)**
    - Shows libraries listed as dependencies but not actually used.

Workbook Exercises

1. **List All Shared Libraries**
    - Inspect `/lib`, `/usr/lib`, and `/usr/local/lib`.
    - Observe versioned vs. unversioned symbolic links (e.g., `libm.so.6 → libm-2.31.so`).
2. **Update Library Cache**
    - Create a directory `/opt/customlib` and put a dummy `.so` file (or symbolic link) there.
    - Add `/opt/customlib` to `/etc/ld.so.conf.d/custom.conf`.
    - Run `sudo ldconfig -v` and verify the new library is recognized (`ldconfig -p | grep customlib`).
3. **Use `LD_LIBRARY_PATH`**
    - Temporarily set `LD_LIBRARY_PATH=/opt/customlib`.
    - Run an executable depending on the custom library.
    - Confirm it finds the library without editing `/etc/ld.so.conf`.
4. **Check Dependencies**
    - Use `ldd /bin/ls` to see the libraries it requires.
    - Use `ldd` on a custom binary if available.
    - (Optional) Try the `-u` option to see if any direct dependencies are unused.
5. **Investigate a Broken App**

- Intentionally remove or rename a `.so` file that an application needs (e.g., `mv libXYZ.so.1 libXYZ.so.1.bak`).
- Attempt to run the application and note the error.
- Restore the file or fix the library path to resolve the error.

Summary

- Linux uses **shared libraries** (`.so`) to avoid embedding common code in each executable, saving resources.
- The **dynamic linker** finds libraries via paths defined in `/etc/ld.so.conf` (and sub-files in `ld.so.conf.d`) and updates a cache with `ldconfig`.
- `LD_LIBRARY_PATH` can override these directories temporarily for testing or specialized setups.
- Tools like `ldd` help identify which libraries an executable (or another library) needs, aiding in troubleshooting.

102.4 Use Debian Package Management

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 102.4**
- **Weight:** 3

Key Knowledge Areas

- Installing, upgrading, and uninstalling Debian binary packages.
- Finding packages containing specific files or libraries (installed or not).
- Obtaining package information (version, contents, dependencies, integrity, status).
- Awareness of `apt` and related commands.

Important Files, Terms, and Utilities

- `/etc/apt/sources.list` (and `/etc/apt/sources.list.d/`) – repository lists
- `dpkg` – the low-level Debian package tool
- `dpkg-reconfigure` – re-run configuration scripts for installed packages
- `apt-get` (or `apt`) – higher-level tool for package handling
- `apt-cache` (or `apt search/show`) – searching in and displaying details about packages

Lesson Overview

In Debian-based Linux distributions (including Ubuntu and others), packages come in `.deb` format. The `dpkg` utility can install and remove `.deb` files, but does not automatically handle dependencies. For that, tools like `apt-get` (or the more modern `apt`) help resolve dependencies, perform upgrades, and search repositories.

1. Using `dpkg` (Debian Package Tool)

1. **Install a `.deb` Package**

```
sudo dpkg -i PACKAGE_FILE.deb
```

   - Installs or upgrades the package if an older version is detected.
   - Fails if dependencies are missing.

2. **Remove a Package**

```
sudo dpkg -r PACKAGE_NAME
```

   - Leaves config files behind.
   - `-P` (purge) removes config files as well.

3. **Listing Installed Packages**

```
dpkg --get-selections
```

   - Outputs every installed package.

4. **Package Contents**

```
dpkg -L PACKAGE_NAME
```

   - Lists all files installed by that package.

5. **Which Package Owns a File?**

```
dpkg-query -S /path/to/file
```

   - Shows the package name that installed the file.

6. **Inspect a `.deb` File**

```
dpkg -I PACKAGE_FILE.deb
```

   - Prints metadata (dependencies, maintainer, version, etc.).

7. **Reconfigure Installed Packages**

```
sudo dpkg-reconfigure PACKAGE_NAME
```

   - Reruns post-install scripts, can fix or reset configuration.

Note. *Using `-force` overrides safety checks but risks breaking the system.*

2. `apt-get` or `apt` for Dependency Handling

1. **Updating Package Index**

```
sudo apt-get update
```

   - Fetches latest package info from repositories.

2. **Installing Packages**

```
sudo apt-get install PACKAGE_NAME
```

   - Resolves and installs dependencies automatically.

3. **Removing Packages**

```
sudo apt-get remove PACKAGE_NAME
```

   - Leaves config files; use `-purge` to remove them.

4. **Fixing Broken Dependencies**

```
sudo apt-get install -f
```

   - Attempts to fix unmet dependencies.

5. **Upgrading Packages**

```
sudo apt-get upgrade
```

   - Upgrades all installed packages to latest versions in the repositories.
   - Run `apt-get update` beforehand to refresh index.

6. **Cleaning Cache**

```
sudo apt-get clean
```

   - Clears `.deb` files in `/var/cache/apt/archives` to free space.

3. Searching for Packages

1. `apt-cache search` (**or** `apt search`)

```
apt-cache search KEYWORD
```

   - Lists packages whose name/description match `KEYWORD`.

2. `apt-cache show` (**or** `apt show`)

```
apt-cache show PACKAGE_NAME
```

   - Provides detailed info (dependencies, version, maintainers, etc.).

3. `apt-file`

   - May need `sudo apt-get install apt-file` first.
   - Then `sudo apt-file update` to sync its own index.
   - **Listing contents of a package:**

```
apt-file list PACKAGE_NAME
```

   - **Finding which package provides a file:**

```
apt-file search FILENAME
```

   - Unlike `dpkg-query -S`, works for **uninstalled** packages as well.

4. Configuring Repositories (`sources.list`)

- `/etc/apt/sources.list` or `/etc/apt/sources.list.d/*.list`
- Lines typically look like:

```
deb http://deb.debian.org/debian buster main contrib non-free
deb-src http://deb.debian.org/debian buster main contrib non-free
```

- **Archive types:** `deb` (binary packages) or `deb-src` (source).
- **Distributions:** e.g., `buster`, `stable`, `testing`, or codenames for Ubuntu.
- **Components:** `main`, `contrib`, `non-free`, `universe`, `multiverse`, etc.

After editing sources, run:

```
sudo apt-get update
```

Workbook Exercises

1. **Install a `.deb` File with `dpkg`**
   - Download a `.deb` (e.g., from a website).
   - Try to install:

     ```
     sudo dpkg -i package.deb
     ```

   - If dependencies fail, note the error message. Then fix them using either `dpkg` again or `apt-get install -f`.
2. **Purge an Installed Package**
   - Select a small package to remove:

     ```
     sudo apt-get remove --purge PACKAGE_NAME
     ```

   - Confirm config files are removed by checking `dpkg -L PACKAGE_NAME` (should say not installed).
3. **Reconfigure a Package**
   - Example:

     ```
     sudo dpkg-reconfigure tzdata
     ```

   - Verify you can reset or change the time zone.
4. **Search and Install with `apt`**
   - Run:

     ```
     apt-cache search KEYWORD
     ```

   - Pick a package from the results and install it with `apt-get install`.
   - Check the installed files with:

     ```
     dpkg -L PACKAGE_NAME
     ```

5. **Repository Configuration**
   - Inspect `/etc/apt/sources.list` and `/etc/apt/sources.list.d/`.
   - Optionally add a new repository line (e.g., a backports line).
   - Run `sudo apt-get update` and check if new packages are available.
6. **List a Package's Contents**
   - Install `apt-file` if needed:

```
sudo apt-get install apt-file
sudo apt-file update
```

- List contents for a known package:

```
apt-file list PACKAGE_NAME
```

- Search for a file across all packages:

```
apt-file search /bin/somefile
```

Summary

- `dpkg` handles `.deb` packages at a low level but does **not** resolve dependencies automatically.
- `apt-get`, `apt`, and `apt-cache` provide higher-level features like dependency resolution, searching repositories, and automated upgrades.
- `apt-file` allows searching within packages (even those not installed).
- The `sources.list` (and `.list` files in `/etc/apt/sources.list.d`) specify where apt should look for packages.
- Knowing these tools is critical for effectively installing, upgrading, or removing software in Debian-based systems, aligning with the LPIC-1 **102.4** objective.

102.5 Use RPM and YUM Package Management

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 102.5**
- **Weight:** 3

Key Knowledge Areas

- Installing, re-installing, upgrading, and removing packages with **rpm**, **YUM**, and **Zypper**
- Obtaining information on RPM packages (version, dependencies, signatures, etc.)
- Determining the files a package provides, and finding which package a specific file comes from
- Awareness of **dnf** (successor to YUM in Fedora-based systems)

Important Files, Terms, and Utilities

- **rpm**, **rpm2cpio**
- `/etc/yum.conf`, `/etc/yum.repos.d/`
- **yum**, **zypper**, **dnf**
- Various `.repo` configuration files

Lesson Overview

Linux distributions derived from Red Hat (RHEL, Fedora, CentOS, openSUSE) typically use RPM (`.rpm` files) for package distribution. The **rpm** utility handles low-level package operations but does **not** resolve dependencies automatically. Higher-level tools like **yum**, **dnf**, and **zypper** manage dependencies, perform system upgrades, and handle repository configurations.

1. Managing Packages with `rpm`

1. **Installing a Package**

```
rpm -ivh PACKAGE_FILE.rpm
```

   - **-i:** install
   - **-v:** verbose

- **-h:** show progress with hash marks

2. **Upgrading a Package**

```
rpm -Uvh PACKAGE_FILE.rpm
```

   - Installs if not already present; upgrades if older version is detected.
   - **-F:** freshen (upgrade only if installed; skip if not).

3. **Removing (Erasing) a Package**

```
rpm -e PACKAGE_NAME
```

   - Fails if other packages depend on it.
   - Remove those dependents first or specify them all at once.

4. **Querying Installed Packages**

   - **List all packages:**

```
rpm -qa
```

   - **Query a package's info:**

```
rpm -qi PACKAGE_NAME
```

   - **List files in a package:**

```
rpm -ql PACKAGE_NAME
```

   - **Find which package owns a file:**

```
rpm -qf /path/to/file
```

5. **Inspecting an Uninstalled Package**

   - **Metadata (info):**

```
rpm -qip PACKAGE_FILE.rpm
```

   - **Contents (file list):**

```
rpm -qlp PACKAGE_FILE.rpm
```

6. **Dependencies**

   - **rpm** will list missing dependencies but cannot automatically resolve them.
   - Use **yum**, **dnf**, or **zypper** to handle dependencies more effectively.

2. YUM (YellowDog Updater Modified)

1. **Searching for Packages**

```
yum search KEYWORD
```

   - Searches names and summaries for `KEYWORD`.

2. **Installing a Package**

```
yum install PACKAGE_NAME
```

   - Resolves and installs dependencies automatically.

3. **Removing a Package**

```
yum remove PACKAGE_NAME
```

   - Also removes packages that depend on it.

4. **Upgrading Packages**

```
yum update PACKAGE_NAME
```

   - Without a package name, updates the entire system.

5. **Checking for Updates**

```
yum check-update [PACKAGE_NAME]
```

   - Lists available updates; omit package name to check all installed packages.

6. **Which Package Provides a File**

```
yum whatprovides FILENAME
```

   - Helps identify the package that contains a needed file or library.

7. **Getting Package Info**

```
yum info PACKAGE_NAME
```

   - Shows version, architecture, summary, repo source, etc.

8. **Repositories** (`/etc/yum.repos.d/*.repo`)

- **Add/Remove Repos:** `yum-config-manager -add-repo URL` / `yum-config-manager -remove-repo REPO_ID`
- **Enable/Disable Repos:** `yum-config-manager -enable REPO_ID` / `yum-config-manager -disable REPO_ID`
- **List Repos:** `yum repolist all`

9. **Cleaning Cache**

```
yum clean [packages|metadata|all]
```

- Frees disk space by removing cached `.rpm` files or metadata.

3. DNF (Dandified YUM)

1. **Overview**
   - Used by Fedora and newer Red Hat-based systems.
   - Similar commands to **yum**.
2. **Basic Commands**
   - **Search:** `dnf search KEYWORD`
   - **Install:** `dnf install PACKAGE_NAME`
   - **Remove:** `dnf remove PACKAGE_NAME`
   - **Upgrade:** `dnf upgrade [PACKAGE_NAME]` (upgrade entire system if no package specified)
   - **Which Package Provides a File:** `dnf provides /path/to/file`
   - **List Installed Packages:** `dnf list -installed`
3. **Repositories**
   - **List all:** `dnf repolist [-enabled|-disabled]`
   - **Add:** `dnf config-manager -add-repo URL`
   - **Enable/Disable:** `dnf config-manager -set-enabled REPO_ID` / `dnf config-manager -set-disabled REPO_ID`
4. **Cleaning Cache**

```
dnf clean all
```

- Removes cache data (packages, metadata).

4. Zypper (openSUSE / SUSE)

1. **Refreshing Repositories**

```
zypper refresh
```

   - Updates repository metadata.

2. **Searching for Packages**

```
zypper search [--installed-only|--not-installed|--provides /file]
```

   - `zypper se KEYWORD`
   - `zypper se -i KEYWORD` (installed only)
   - `zypper se -provides /path/to/file` (find package providing a file)

3. **Installing Packages**

```
zypper install PACKAGE_NAME
```

   - Or `zypper in PACKAGE_NAME`.

4. **Upgrading Packages**

```
zypper update [PACKAGE_NAME]
```

   - Without specifying a package, updates all.

5. **Removing Packages**

```
zypper remove PACKAGE_NAME
```

   - Or `zypper rm PACKAGE_NAME`.

6. **Package Info**

```
zypper info PACKAGE_NAME
```

   - Shows version, repository, summary, etc.

7. **Listing Package Contents**

```
zypper search --provides /path/to/file
```

   - Or `zypper info -requires PACKAGE_NAME` for dependencies.

8. **Repositories**

   - **List:** `zypper repos`
   - **Add:** `zypper addrepo URL ALIAS`
   - **Remove:** `zypper removerepo ALIAS`
   - **Enable/Disable:**

```
zypper modifyrepo -e ALIAS # enable
zypper modifyrepo -d ALIAS # disable
```

   - **Auto-Refresh:**

```
zypper modifyrepo -f ALIAS # enable auto-refresh
zypper modifyrepo -F ALIAS # disable auto-refresh
```

Workbook Exercises

1. **Basic `rpm` Operations**

   - Download an `.rpm` package (e.g., `wget http://example.com/somepackage.rpm`).
   - Install it via:

```
sudo rpm -ivh somepackage.rpm
```

   - Query what files it installed (`rpm -ql PACKAGE_NAME`).
   - Remove it (`rpm -e PACKAGE_NAME`).

2. **Resolve Dependencies with YUM**

   - Try installing a package that requires another package.
   - Notice that `yum` automatically pulls needed dependencies.
   - Remove the newly installed package and dependencies if desired:

```
sudo yum remove PACKAGE_NAME
```

3. **Which Package Owns a File?**

   - Use `yum whatprovides /usr/bin/zipinfo` (or a similar file) to see who owns it.
   - Confirm with `rpm -qf /usr/bin/zipinfo`.

4. **Update the Entire System**

   - On a CentOS or RHEL system, run:

```
sudo yum update
```

- Reboot if a new kernel is installed.

5. **Add/Enable a New Repository**
   - For CentOS, add a repo:

```
yum-config-manager --add-repo https://example.com/custom.repo
```

   - Use `yum repolist all` to confirm it appears, then enable if needed.

6. **Zypper Install**
   - On an openSUSE system, run:

```
sudo zypper refresh
sudo zypper search unzip
sudo zypper install unzip
```

   - Check the installed files via `rpm -ql unzip` or `zypper info unzip`.

7. **dnf Operations**
   - On a Fedora system, search for `gimp`:

```
dnf search gimp
```

   - Install it:

```
dnf install gimp
```

   - Remove it:

```
dnf remove gimp
```

Summary

- **rpm** is the low-level tool for installing `.rpm` packages, but it does **not** handle dependencies automatically.
- **yum**, **dnf**, and **zypper** provide higher-level package management with automatic dependency resolution, repository management, and system-wide updates.
- Each tool has commands for searching packages, installing, upgrading, removing, and listing package contents.
- Understanding these utilities is critical for effectively managing software on RPM-based Linux distributions—an important skill for LPIC-1 certification and real-world administration.