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 $\geq$ RAM if hibernation is used).

3. LVM (Logical Volume Manager)

1. **Overview**
   - Provides flexible "virtual" partitions called **Logical Volumes (LVs)**.
   - **Physical Volumes (PVs)** $\rightarrow$ grouped into **Volume Groups (VGs)** $\rightarrow$ 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.

Multiple-Choice Questions for 102.1

1. Which statement best describes the purpose of a **partition table** on a disk?

   A) It is the directory on the filesystem that contains user data

   B) It is the bootloader used to start the operating system

   C) It contains information about the sectors and types of partitions on the disk

   D) It is the tool used to create logical volumes

2. On most Linux distributions, **removable media** (e.g., USB drives, memory cards) are:

   A) Automatically mounted under `/media/USER/LABEL`

   B) Expected to be manually mounted under `/opt`

   C) Required to be unmounted only after a reboot

   D) Limited to read-only access by default

3. What is one **benefit** of creating a dedicated `/boot` partition?

   A) It automatically encrypts the disk to improve security

   B) It merges multiple storage devices into a single volume

   C) It allows `/home` to remain untouched during upgrades

   D) It ensures the system can still boot if the root filesystem is corrupted

4. Which of the following is a **directory** that frequently benefits from being put on a separate partition, due to potential growth of log files and database data?

   A) `/bin`

   B) `/var`

   C) `/root`

   D) `/tmp`

5. The **EFI System Partition (ESP)** on a UEFI-based system is typically:

   A) Formatted with an ext4 filesystem

   B) Formatted with a FAT-based filesystem

   C) Placed at the end of the disk to maximize disk space

   D) Labeled as `/swap` in the `/etc/fstab` file

6. What is the primary **purpose** of the **swap partition**?

   A) To store user home directories

   B) To store kernel bootloader files for GRUB

   C) To hold system logs and database files

   D) To provide virtual memory (swap out pages from RAM)

7. Which of the following is **NOT** a reason to create a separate `/home` partition?

   A) To prevent boot issues on legacy BIOS systems

   B) To simplify system reinstallation while keeping user data

   C) To allow for different filesystem choices for user data

   D) To avoid overwriting user data during an OS upgrade

8. A **mount point** is best described as:

   A) A command used to set the filesystem read-only

   B) A method to copy data from one partition to another

   C) A directory where a filesystem is attached to the directory tree

   D) A device driver for SATA disks

9. Which directory commonly holds the **Apache Web Server** data by default?

   A) `/home/apache`

   B) `/var/www/html`

   C) `/srv/httpd`

   D) `/apache/www`

10. Under **Logical Volume Management (LVM)**, which statement is correct about **Volume Groups (VGs)**?

    A) VGs cannot be reduced in size after creation

    B) VGs are used only for encrypting partitions

    C) VGs aggregate multiple physical volumes into one large pool of storage

    D) VGs must reside on a separate `/vg` partition

11. Which of the following directories is typically **not** located under `/home` on a Linux system?

    A) `/root` (the home directory for the root user)

    B) `/john` (a normal user's home directory)

    C) `/jack` (a normal user's home directory)

    D) `/carol` (a normal user's home directory)

12. Which **bootloader** is most commonly used on modern Linux systems for loading the operating system?

    A) syslinux

    B) LILO

    C) rEFIt

    D) GRUB2

13. The EFI System Partition (ESP) is generally **mounted** under which directory on a Linux system?

    A) `/boot`

    B) `/ESP`

    C) `/boot/efi`

    D) `/usr/local/esp`

14. Which directory contains "variable data" and is known to grow due to logs and other services?

    A) `/etc`

    B) `/var`

    C) `/mnt`

    D) `/bin`

15. Which of the following statements correctly describes **Physical Volumes (PVs)** in LVM?

    A) They are the mount points for the swap partition

    B) They are user directories that can be encrypted

    C) They are used only for backing up the master boot record

    D) They are block devices (partitions, disks) that become part of a volume group

16. Why might a system administrator place `/home` and `/var` on **separate physical disks**?

    A) To reduce the impact of one disk's failure on another

    B) To comply with GPT partition ID requirements

    C) To ensure that `/boot` remains fully accessible

    D) To use a unique journaling filesystem only on `/home`

17. **Logical Volumes (LVs)** in an LVM setup appear to the operating system as:

    A) Symbolic links pointing to `/boot`

    B) BIOS-level CHS addresses

    C) A directory tree with unlimited capacity

    D) Normal block devices (e.g., `/dev/VGNAME/LVNAME`)

18. If you need to **manually** mount an additional filesystem that is **not** removable media, the best practice is typically to mount it under which directory?

    A) `/var`

    B) `/home`

    C) `/mnt`

    D) `/opt/extra`

19. According to common recommendations, placing system logs and database files under a dedicated `/var` partition helps:

    A) Prevent swap space from being used

    B) Protect the root filesystem if `/var` fills up

    C) Force all logs to be read-only

    D) Make the system boot faster

20. Which of the following describes a **swap file**?

    A) A regular file residing on an existing filesystem, configured to provide additional swap space

    B) A partition created at the beginning of the disk and labeled as `0xEF`

    C) A mount point used only for kernel boot files

    D) A special utility run by the BIOS to load the operating system

Fill-in-the-Blank Questions for 102.1

1. The folder where user data files and preferences are typically stored is ___________.
2. The BIOS limitation of 1024 cylinders initially required a dedicated ___________ partition to be placed at the beginning of the disk for bootloader files.
3. A ___________ is the logical subdivision of a physical disk, storing data separately from other subdivisions on the same disk.
4. On systems with UEFI firmware, the ___________ partition stores boot loader and kernel files in a FAT-based filesystem.
5. The command that prepares a partition for use as swap space is ___________.
6. The directory ___________ is used for "variable data" such as logs, database files, caches, and spool directories.
7. A filesystem must be "attached" to the system via a process called ___________ before you can access its contents.
8. In LVM, multiple Physical Volumes can be combined to form a larger ___________.
9. A ___________ is a file that can serve as additional swap space without requiring a dedicated swap partition.
10. Traditional partitioning can be inflexible when allocating disk space. One way to overcome this limitation is by using ___________.

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.

Multiple-Choice Questions for 102.2

1. Which of the following statements correctly describes the purpose of the Master Boot Record (MBR)?

   A) It is a legacy BIOS setting used to store GPU firmware information.

   B) It is a reserved partition for the `/boot` directory on GPT disks.

   C) It contains a high-level overview of the Linux file system hierarchy.

   D) It contains a partition table and minimal bootstrap code used to start the boot process.

2. Which command is commonly used to generate a new GRUB 2 configuration file?

   A) `grub-init /dev/sda`

   B) `update-grub` (or `grub-mkconfig -o /boot/grub/grub.cfg`)

   C) `mkconfig -g`

   D) `grub-legacy -o /boot/grub/menu.lst`

3. In GRUB 2, which file is recommended for manual editing to permanently change bootloader settings?

   A) `/boot/grub/menu.lst`

   B) `/etc/grub.d/custom.cfg`

   C) `/etc/default/grub`

   D) `/boot/grub/grub.cfg`

4. Which directive in `/etc/default/grub` allows the last chosen boot option to become the default on the next reboot?

   A) `GRUB_SAVEDEFAULT=`

   B) `GRUB_ENABLE_CRYPTODISK=`

   C) `GRUB_TIMEOUT=`

   D) `GRUB_CMDLINE_LINUX=`

5. What is the main purpose of an initial RAM disk (initrd)?

   A) It is another name for the GRUB 2 configuration file.

   B) It is a dedicated swap partition to store kernel crash dumps.

   C) It compresses the kernel to reduce its size in `/boot`.

   D) It provides a minimal root filesystem so the kernel can mount the real root filesystem.

6. Which GRUB 2 command is used in a manual (interactive) session to load the Linux kernel?

   A) `linux`

   B) `initrd`

   C) `chainloader`

   D) `set root=`

7. Which of the following is **not** true regarding GPT?

   A) It is part of the UEFI specification.

   B) It is incompatible with BIOS-based systems.

   C) It supports disks larger than 2 TB.

   D) It uses GUIDs to identify partitions.

8. When chainloading Windows from GRUB Legacy or GRUB 2, which command loads the Windows bootloader from the first sector of its partition?

   A) `initrd /windows.img`

   B) `search -set=root -fs-uuid <uuid>`

   C) `chainloader +1`

   D) `module /boot/grub/i386-pc/chain.mod`

9. In GRUB Legacy, which file typically stores the boot menu configuration?

   A) `/boot/grub/menu.lst`

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

   C) `/etc/default/grub`

   D) `/etc/grub.d/menu.cfg`

10. You are using a Live CD to rescue a system. You have mounted the dedicated boot partition at `/mnt/bootpartition`. Which command correctly installs GRUB 2 to the MBR on `/dev/sda`?

   A) `grub-install -root-directory=/mnt/bootpartition /dev/sda`

   B) `update-grub -d /dev/sda /mnt/bootpartition`

   C) `grub-mkconfig -o /dev/sda`

   D) `grub-install -boot-directory=/mnt/bootpartition /dev/sda`

11. In the context of GRUB 2, which file is **not** recommended for direct editing?

   A) `/etc/grub.d/40_custom`

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

   C) `/boot/grub/fonts/unicode.pf2`

   D) `/etc/default/grub`

12. Which of the following describes a scenario where a separate `/boot` partition is particularly useful?

   A) When you want to mount `/boot` as a swap partition.

   B) When you plan to store the entire root filesystem on MBR with no separate partitions.

   C) When you want all boot files in `/bin/boot` for convenience.

   D) When you use encryption or compression methods not supported by the bootloader.

13. Which GRUB 2 command can help you identify partitions and disks from the GRUB shell?

   A) `ls`

   B) `disklist`

   C) `list-devices`

   D) `find /boot/grub/stage1`

14. What is the typical name of the Linux kernel file found in `/boot`?

   A) `System.map-VERSION`

   B) `initrd.img-VERSION`

   C) `vmlinuz-VERSION`

   D) `config-VERSION`

15. Which of the following lines in a GRUB 2 `menuentry` block is responsible for loading the initial RAM disk?

    A) `chainloader +1`

    B) `initrd /initrd.img`

    C) `set root=(hd0,1)`

    D) `linux /vmlinuz root=/dev/sda1 ro`

16. When using GRUB Legacy from a bootable rescue image, which command installs GRUB to the MBR **after** setting the correct root device?

    A) `update-grub (hd0)`

    B) `mkconfig -o (hd0)`

    C) `install (hd0,msdos1)`

    D) `setup (hd0)`

17. Which line in `/etc/default/grub` disables the boot menu countdown and waits indefinitely for user input?

    A) `GRUB_DEFAULT=saved`

    B) `GRUB_ENABLE_CRYPTODISK=y`

    C) `GRUB_TIMEOUT=-1`

    D) `GRUB_CMDLINE_LINUX_DEFAULT=""`

18. Which component of the Linux filesystem organizes kernel symbols (variables, functions) with their memory addresses for debugging?

    A) `vmlinuz-VERSION`

    B) `config-VERSION`

    C) `initrd-VERSION`

    D) `System.map-VERSION`

19. Which GRUB 2 command would you use within the shell to specify the disk or partition containing the root filesystem?

    A) `chainloader +1`

    B) `set root=(hd0,1)`

    C) `root (hd0,0)`

    D) `initrd /initrd.img`

20. If a misconfiguration causes GRUB 2 to drop into a rescue shell (`grub rescue>`), which modules often need to be loaded manually to regain the normal GRUB shell functionality?

    A) `normal` and `linux`

    B) `fs_uuid` and `configfile`

    C) `chainloader` and `search`

    D) `ls` and `initrd`

Fill-in-the-Blank Questions for 102.2

1. When a computer is powered on, the first software component to run is the ___________.
2. On a BIOS system using MBR partitioning, the very first 512-byte sector of the disk is called the ___________.
3. The Linux kernel is typically stored in a compressed file named ___________-VERSION in the '/boot' directory.
4. The minimal root filesystem required to load the real root filesystem is contained in the ___________ file.
5. The file ___________-VERSION contains a look-up table used for debugging kernel panics.
6. In GRUB 2, manual edits to the bootloader settings should be done in ___________, rather than directly editing '/boot/grub/grub.cfg'.
7. The ___________ command scans available disks and partitions to build a list of operating systems for GRUB 2.
8. A dedicated ___________ partition is often used to store all files needed for the boot process separately from the root filesystem.
9. GRUB Legacy stores its main configuration in the file ___________.
10. A technique known as ___________ is used by GRUB to load other operating systems' bootloaders (e.g., Windows) from their first sector.

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.

Multiple-Choice Questions for 102.3

1. What is the purpose of having a **dynamic** library as opposed to a **static** one?

   A) It embeds all external library code into the final executable.

   B) It decreases the chance of version conflicts between libraries.

   C) It simplifies the debugging process by combining all symbols in one file.

   D) It allows multiple programs to share the same library file in memory at runtime.

2. Which directory is **commonly** used by Linux systems to store 64-bit libraries?

   A) `/etc/x86_64-linux-gnu`

   B) `/lib64`

   C) `/bin64`

   D) `/usr/local/opt/lib64`

3. When you install a new library in a custom location, which **environment variable** can you set to let the system know about the additional library path?

   A) `LIB_EXTRA_PATH`

   B) `LD_DEBUG_PATH`

   C) `LD_LIBRARY_PATH`

   D) `PATHLIB`

4. After adding a new `.conf` file under `/etc/ld.so.conf.d/`, which **command** should you usually run to ensure the changes take effect?

   A) `sudo ldconfig`

   B) `echo $LD_LIBRARY_PATH`

   C) `ldd -u /etc/ld.so.conf`

   D) `touch /etc/ld.so.cache`

5. A **static library** is characterized by:

   A) Having no effect on program size.

   B) Being dynamically loaded at runtime.

   C) Residing exclusively under `/usr/local/lib`.

   D) Being fully integrated into the binary during link time.

6. Which file typically holds **symbolic links** to the actual versioned shared library files and speeds up library lookups?

   A) `/etc/bash.bashrc`

   B) `/etc/ld.so.conf.d/`

   C) `/etc/ld.so.cache`

   D) `/etc/profile`

7. What is the **main** function of the command `ldd /usr/bin/git`?

   A) It loads any missing libraries into memory for the executable.

   B) It displays which shared libraries and memory addresses the program will use.

C) It modifies `/etc/ld.so.conf` to remove outdated references.

D) It compiles the binary from source code.

8. Which statement is **true** regarding the file `/etc/ld.so.conf` on many modern Linux distributions?

   A) It directly lists all directories containing library files without any inclusion mechanisms.

   B) It typically has an `include` line that references additional `.conf` files in `/etc/ld.so.conf.d/`.

   C) It is unrelated to dynamic library configuration.

   D) It must only contain symbolic links, not absolute paths.

9. Which command option would show **unused** direct library dependencies for an executable?

   A) `ldconfig -v`

   B) `ldconfig -p`

   C) `ldd -verbose`

   D) `ldd -u`

10. The **logical name** given to a shared library (like `libm.so.6`) is called the:

    A) Full path reference.

    B) Statically linked file name.

    C) Base library handle.

    D) Soname.

11. Which directory is **not** typically part of the default library search path?

    A) `/opt/libraries/`

    B) `/lib/`

    C) `/usr/lib/`

    D) `/usr/local/lib/`

12. In a typical modern Linux system, if you run `ldconfig -p`, what does the `-p` option do?

    A) It permanently deletes old cache entries from `/etc/ld.so.cache`.

    B) It lists the directories and candidate libraries in the **current** cache.

    C) It prints out any undefined symbols in the loaded libraries.

    D) It prompts the user for additional library paths.

13. During the build process, an executable may mark certain libraries as **NEEDED** even if they aren't used at runtime. This often happens because of:

    A) Accidental corruption in the library file.

    B) Missing environment variables in the developer's system.

    C) Linker flags that reference multiple libraries.

    D) Repeated calls to `ldconfig`.

14. Which of the following **best** describes what `ld-linux.so` (or `ld.so`) does?

    A) It locates, loads, and links the needed shared libraries at runtime.

    B) It compiles source code into object files for linking.

    C) It performs static linking of libraries during application build time.

D) It generates symbolic links in `/etc/ld.so.conf.d/` automatically.

15. What is a **key advantage** of using shared libraries on a system with many processes?

    A) Each application runs in its own memory space without any shared code.

    B) You never need to update libraries since they are compiled into each executable.

    C) They can be loaded in user space without root privileges.

    D) Only one copy of the library is loaded into memory and used by multiple processes.

16. If a program is **statically** linked against a library, then:

    A) You do **not** need the library on the system at runtime for the program to function.

    B) You must always place a copy of the library in `/lib/x86_64-linux-gnu`.

    C) The linker will load the library into memory each time it's called.

    D) There must be an exact match between library version and kernel version.

17. What does the command `ldconfig -v` do **in addition** to updating the library cache?

    A) It permanently locks the library version in place.

    B) It displays verbose details about the libraries found, links created, and directories scanned.

    C) It only updates symbolic links, but not the cache.

    D) It filters out symbolic links that are unused or broken.

18. Which of these files would you **most likely** edit or create to specify a custom library directory (e.g., `/opt/mylib`)?

    A) `/etc/ld.so.conf.d/custom.conf`

    B) `/usr/local/lib/custom.ld.so.conf`

    C) `/var/run/ldconfig/ld.so.conf`

    D) `/etc/bash.bashrc`

19. `LD_LIBRARY_PATH` is similar to `PATH` in the sense that:

    A) Both are used solely for system administrators to track dependencies.

    B) Both contain hashed references to libraries that get pre-loaded.

    C) Both are environment variables that list directories to search, but for different purposes (executables vs. libraries).

    D) Neither can be exported in a user shell.

20. Which statement accurately **describes** the role of symbolic links like `libpthread.so.0 -> libpthread-2.31.so`?

    A) They are stored in `/etc/ld.so.cache` for quick loading of kernel modules.

    B) They connect older kernel releases to new libraries.

    C) They enable the system to reference a library by its **soname** while pointing to the actual versioned file.

    D) They are used exclusively for statically linking code.

Fill-in-the-Blank Questions for 102.3

1. The utility used to check the **shared libraries** required by a specific program is ___________.
2. The **environment variable** that can be set to add or override library paths at runtime is ___________.
3. On many Linux systems, the file '/etc/ld.so.conf' includes a line pointing to configuration files in the ___________ directory.
4. When building an application, if the library code is **copied** into the executable at link time, we say the program is using ___________ linking.
5. If we install a new library in '/usr/local/mylib' and do **not** want to modify system-wide configurations, we can temporarily set ___________=/usr/local/mylib.
6. A library name like 'libfuse.so.2' is often a symbolic link pointing to a **versioned** file such as 'libfuse.so.2.9.7'; this more general filename is often called the ___________.
7. The command ___________ -p lists the directories and candidate libraries stored in the current library cache.
8. We typically run ___________ after modifying or adding a new '.conf' file under '/etc/ld.so.conf.d/' to update the library cache.
9. The file '/etc/ld.so.cache' is used to **speed up** the lookup of ___________.
10. A static library file typically ends in the extension ___________.

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.

Multiple-Choice Questions for 102.4

1. Which parameter in `dpkg` is used to remove both a package and its configuration files?

   A) `-r`

   B) `-I`

   C) `-S`

   D) `-P`

2. Which of the following commands updates the local package index using APT?

   A) `apt-get remove`

   B) `apt-get install -f`

   C) `apt-get update`

   D) `apt-get purge`

3. Which `dpkg` command option allows you to list all files that a package has installed on the system?

   A) `dpkg -get-selections`

   B) `dpkg -I`

   C) `dpkg-reconfigure`

   D) `dpkg -L`

4. Which command is used to remove a package but keep its configuration files?

   A) `dpkg -P`

   B) `apt-get remove`

   C) `apt-get install -f`

   D) `apt-get purge`

5. What is the correct `Archive type` that indicates a repository contains ready-to-run packages?

   A) `deb-src`

   B) `main`

   C) `deb`

   D) `contrib`

6. What is the default location of the local cache where `.deb` files are downloaded before installation?

   A) `/etc/apt/sources.list`

   B) `/var/cache/apt/archives`

   C) `/usr/local/cache/dpkg`

   D) `/var/dpkg/archives/partial`

7. Which command helps you restore or re-run the initial configuration process of a package?

   A) `dpkg -get-selections`

   B) `dpkg -L`

   C) `dpkg-reconfigure`

   D) `dpkg -S`

8. Which `dpkg` option can show you which package owns a specific file on the filesystem (e.g., `/usr/bin/example`)?

   A) `dpkg -I`

   B) `dpkg -L`

   C) `dpkg -P`

   D) `dpkg -S`

9. Which of the following statements is true about `apt-get install -f`?

   A) It attempts to fix broken dependencies by installing missing packages.

   B) It removes all configuration files of broken packages.

   C) It removes all broken packages from the system.

   D) It upgrades all packages to the latest version.

10. Which command can be used to search for a package by a keyword in the APT package index?

    A) `apt-get show`

    B) `apt-cache search`

    C) `dpkg -L`

    D) `dpkg-query -S`

11. Which parameter of `dpkg` lists the basic metadata (like version, architecture, dependencies) of a `.deb` package file?

    A) `-I`

    B) `-r`

    C) `-P`

    D) `-L`

12. Which of these lines in `/etc/apt/sources.list` indicates a repository of source packages rather than binary packages?

    A) `deb-src`

    B) `deb http://repo.example.com stable main`

    C) `deb /var/cache/apt/archives stable main`

    D) `deb http://repo.example.com sources main`

13. Which command is used to remove unnecessary `.deb` files in the local cache under `/var/cache/apt/archives`?

    A) `apt-get remove`

    B) `apt-get purge`

    C) `apt-get update`

    D) `apt-get clean`

14. Which `dpkg` parameter performs the same function as `dpkg -r` but leaves configuration files behind?

    A) `-r`

    B) `-P`

    C) `-S`

    D) `-I`

15. Which APT command will remove a package along with its configuration files?

    A) `apt-get remove`

    B) `apt-get update`

    C) `apt-get install -f`

    D) `apt-get purge`

16. When a Debian-based system warns that certain packages are "kept back," which command would you generally use to upgrade them?

    A) `apt-get dist-upgrade`

    B) `apt-file search`

    C) `dpkg -P`

    D) `dpkg -i`

17. Which Debian repository component includes software that is DFSG-compliant but depends on non-free components?

    A) `main`

    B) `contrib`

    C) `multiverse`

    D) `restricted`

18. Which APT utility focuses on searching for package information and displaying metadata about packages?

    A) `dpkg`

    B) `apt-file`

    C) `apt-cache`

    D) `dpkg-query`

19. Which of the following `apt-get` commands will remove a package but leave the configuration files on the system?

    A) `apt-get purge`

    B) `apt-get remove`

    C) `apt-get install -f`

    D) `apt-get upgrade`

20. Which main Debian repository contains packages that are compliant with the Debian Free Software Guidelines (DFSG)?

    A) `restricted`

    B) `non-free`

    C) `main`

    D) `multiverse`

Fill-in-the-Blank Questions for 102.4

1. To list all **installed packages** on a Debian-based system using dpkg, you can run:
   `dpkg -get-`__________.
2. The **Advanced Package Tool**, also known as APT, uses repository information from the file:
   `/etc/apt/`__________`.list`.
3. If you have missing dependencies after a failed install, you can attempt to fix them with:
   `apt-get install` __________.
4. You can use ‘dpkg‘ with the **-I** parameter to get __________ about a .deb package file.
5. The package list that APT uses is also known as the APT __________.
6. The parameter ‘dpkg -L‘ lets you list the __________ installed by a particular package.
7. Lines beginning with a __________ character in ‘/etc/apt/sources.list‘ are ignored because they are comments.
8. The command:
   `apt-`__________ `search p7zip`
   is used to search for a package containing the term “p7zip.”
9. To remove all downloaded package files and reclaim disk space, you run:
   `apt-get` __________.
10. The configuration files are only completely removed when you use the dpkg parameter ‘-P‘, which stands for:
    `dpkg -P` __________.

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.

Multiple-Choice Questions for 102.5

1. Which `rpm` parameter is used to remove (erase) an installed package?

   A) `-U`

   B) `-e`

   C) `-F`

   D) `-qa`

2. Which `rpm` command allows you to query an *uninstalled* package file for information (name, version, etc.)?

   A) `rpm -qi`

   B) `rpm -ql`

   C) `rpm -qa`

   D) `rpm -qip`

3. Which `yum` command installs a package named `vim` from the configured repositories?

   A) `yum install vim`

   B) `yum remove vim`

   C) `yum info vim`

   D) `yum repolist vim`

4. Which `yum` subcommand removes an installed package from your system?

   A) `yum whatprovides`

   B) `yum info`

   C) `yum remove`

   D) `yum repolist`

5. Using `yum`, which command do you run to find the package that provides `/usr/bin/unzip`?

   A) `yum search /usr/bin/unzip`

   B) `yum repolist /usr/bin/unzip`

   C) `yum list installed /usr/bin/unzip`

   D) `yum whatprovides /usr/bin/unzip`

6. Which of the following `rpm` parameters lists *all* installed packages on the system?

   A) `-e`

   B) `-qa`

   C) `-U`

   D) `-ql`

7. What is the main purpose of the `rpm2cpio` utility?

   A) It converts an RPM file into a `.cpio` archive

   B) It lists installed `.cpio` packages

   C) It creates a `.tar.gz` archive from an RPM

   D) It checks package signatures in cpio format

8. Which `rpm` command could forcibly install (ignoring dependencies) a package named `mypkg.rpm`?

   A) `rpm -Uvh -nodeps mypkg.rpm`

   B) `rpm -e mypkg.rpm`

   C) `rpm -ql mypkg.rpm`

   D) `rpm -qa -nodeps mypkg.rpm`

9. Which `dnf` command updates *all* installed packages on the system to their latest versions?

   A) `dnf info`

   B) `dnf remove`

   C) `dnf upgrade`

   D) `dnf list -installed`

10. When using `dnf`, how do you find which package provides `/usr/bin/unzip`?

    A) `dnf list /usr/bin/unzip`

    B) `dnf provides /usr/bin/unzip`

    C) `dnf repoquery -installed /usr/bin/unzip`

    D) `dnf info /usr/bin/unzip`

11. Which `zypper` command lets you install an RPM file on disk (e.g., `/home/user/newpkg.rpm`) while also resolving dependencies from repositories?

    A) `zypper update /home/user/newpkg.rpm`

    B) `zypper refresh /home/user/newpkg.rpm`

    C) `zypper query /home/user/newpkg.rpm`

    D) `zypper in /home/user/newpkg.rpm`

12. Which `zypper` operator should you use to remove a package named `unzip` from your system?

    A) `zypper refresh unzip`

    B) `zypper rm unzip`

    C) `zypper se -i unzip`

    D) `zypper up unzip`

13. Which `zypper` command syntax is used to see which packages provide a specific file, e.g., `/usr/lib64/libgimpui-2.0.s`

    A) `zypper se -provides /usr/lib64/libgimpui-2.0.so.0`

    B) `zypper addrepo -provides /usr/lib64/libgimpui-2.0.so.0`

    C) `zypper info -provides /usr/lib64/libgimpui-2.0.so.0`

    D) `zypper up -provides /usr/lib64/libgimpui-2.0.so.0`

14. Which `zypper` operator refreshes all enabled repositories to get the latest metadata?

    A) `zypper se`

    B) `zypper info`

    C) `zypper rm`

    D) `zypper refresh`

15. If you only want to *list* available updates (without installing them) using `zypper`, which command would you use?

A) `zypper up -list`

B) `zypper se updates`

C) `zypper list-updates`

D) `zypper in -updates-only`

16. How do you disable an existing repository named `repo-non-oss` using `zypper`?

A) `zypper addrepo -d repo-non-oss`

B) `zypper rm repo-non-oss`

C) `zypper se -d repo-non-oss`

D) `zypper modifyrepo -d repo-non-oss`

17. What does the `yum-config-manager -add-repo <URL>` command do?

A) It removes a repository from `/etc/yum.conf`

B) It adds a new `.repo` file in `/etc/yum.repos.d/` based on the specified URL

C) It automatically upgrades `yum` to the latest version

D) It disables all repositories except the one specified

18. Which `dnf` command removes an installed package from your system?

A) `dnf remove PACKAGENAME`

B) `dnf fetch PACKAGENAME`

C) `dnf localinstall PACKAGENAME`

D) `dnf whatprovides PACKAGENAME`

19. Which `yum` command checks if a new version of a package (e.g., `wget`) is available, *without* installing it?

A) `yum whatprovides wget`

B) `yum info wget`

C) `yum check-update wget`

D) `yum clean metadata wget`

20. Which file stores the primary configuration for `yum` by default?

A) `/etc/rpm.conf`

B) `/var/log/yum.conf`

C) `/etc/yum.conf`

D) `/etc/dnf.conf`

Fill-in-the-Blank Questions for 102.5

1. To remove a package using ‘rpm‘, we use:
   `rpm ___________ PACKAGENAME`.
2. On Debian-based systems, the tool analogous to ‘yum‘ (mentioned in the lesson) is:
   ___________.
3. To search for a package with ‘zypper‘, you can use either:
   `zypper ___________ or zypper ___________`.
4. When using ‘dnf‘, the command to uninstall a package named ‘curl‘ is:
   `dnf ___________ curl`.
5. The main configuration file for yum is located at: ___________.
6. On RPM-based systems, the command ‘rpm -qa‘ means “query ___________.”
7. If you want to list all available updates using ‘yum‘ without installing them, you can run:
   `yum ___________`.
8. To view the metadata of the ‘gimp‘ package using ‘zypper‘, type:
   `zypper ___________ gimp`.
9. The tool that is considered a “fork” or newer version of YUM (primarily used in Fedora) is called:
   ___________.
10. To list the files installed by a package named ‘wget‘ using ‘rpm‘, you would use:
    `rpm -___________ wget`.

102.6 Linux as a virtualization guest

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 102.6**
- **Weight:** 1

Key Knowledge Areas

- General concept of virtual machines (VMs) and containers
- Key elements of Infrastructure as a Service (IaaS), such as compute instances, block storage, networking
- Changing Linux-specific system properties when cloning or templating a VM (e.g., host keys, D-Bus machine ID)
- Using system images to deploy VMs, cloud instances, and containers
- Guest drivers and integration features for Linux VMs
- Awareness of **cloud-init** for automated provisioning

Important Files, Terms, and Utilities

- **Virtual machine**, **Linux container**, **application container**
- **Guest drivers** (e.g., Virtio, VirtualBox Guest Additions)
- **SSH host keys**, **D-Bus machine ID**
- **cloud-init**

1. Virtualization Overview

1. **Hypervisor**
    - Software layer allowing multiple **guest** operating systems to run on a single host.
    - Manages physical resources (CPU, memory, storage).
2. **Types of Hypervisors**
    - **Type-1 (Bare-metal):** Runs directly on hardware (e.g., **Xen**, some KVM implementations).
    - **Type-2 (Hosted):** Runs on top of a host OS (e.g., **VirtualBox**).
3. **Common Hypervisors**
    - **Xen** (Type-1, open source).
    - **KVM** (kernel module in Linux; used with **libvirt**, **QEMU**).
    - **VirtualBox** (cross-platform, Type-2).
4. **Migration**
    - **Cold migration:** Move VM when powered off.
    - **Live migration:** Move a running VM to another hypervisor. Useful for maintenance/resiliency.

2. Types of Virtual Machines

1. **Fully Virtualized (Hardware VM)**
    - Guest OS is unmodified and unaware it's virtualized.
    - CPU extensions (Intel VT-x, AMD-V) often required.
2. **Paravirtualized (PVM)**
    - Guest OS is aware it's running in a VM.
    - Uses special drivers for improved performance (e.g., **Virtio** in KVM, Xen drivers).
3. **Hybrid**
    - Fully virtualized OS that uses paravirtualized drivers for I/O performance boosts (disk, network).

3. Guest Drivers and Tools

- **KVM** → **Virtio** drivers for network/storage.
- **VirtualBox** → **Guest Additions** (mounted via ISO).
- Provide near-native performance for I/O operations.

4. Virtual Machine Definition Example (libvirt + KVM)

- `/etc/libvirt/qemu/` contains XML config files describing VMs:
    - Memory, CPUs, disk images, network interfaces, etc.
- Example snippet:

```
<domain type='kvm'>
  <name>rhel8.0</name>
  <memory unit='KiB'>4194304</memory>
  <vcpu>2</vcpu>
  <devices>
    <disk type='file' device='disk'>
      <source file='/var/lib/libvirt/images/rhel8'/>
      <target dev='vda' bus='virtio'/>
    </disk>
    <interface type='network'>
      <source network='default'/>
      <model type='virtio'/>
    </interface>
    ...
  </devices>
</domain>
```

- **Networking** can be NAT-based via `virbr0` or bridged to the host network.

5. VM Disk Storage Formats

1. **QCOW2 (Copy-on-write)**
   - Thin-provisioned (sparse), only consumes physical space for actual data.
   - Can expand up to a max size.
2. **RAW**
   - Pre-allocated, full-size image.
   - Slight performance advantage.
3. **Other Storage Setups**
   - Physical LVM volumes, SAN, NAS, or advanced solutions (oVirt, Red Hat Virtualization).

6. Cloning and Templates

1. **Templates**
   - Pre-built VM images with baseline OS/configuration.
   - Speeds deployment, reduces repetitive setup steps.
2. **Unique System IDs**
   - Must regenerate **SSH host keys**, **D-Bus machine ID** to avoid duplicates.
   - Example to regenerate machine ID:

     ```
     sudo rm -f /etc/machine-id
     sudo dbus-uuidgen --ensure=/etc/machine-id
     ```

7. Cloud Infrastructure (IaaS)

1. **Compute Instances**
   - Providers bill by CPU/memory usage or by instance count/time.
2. **Block Storage**
   - Persistent storage volumes attached to VMs; performance tiers vary by cost.
3. **Networking**
   - Cloud providers offer subnets, routing, firewalls, DNS, or hybrid on-prem/cloud networking (VPN).
4. **Access via SSH**
   - Typically uses key-based authentication.
   - Some providers auto-generate keys or let you upload your own.

8. `cloud-init` for Automated Provisioning

1. `cloud-init`

   - Tool that runs at boot to configure system settings (network, packages, SSH keys, etc.).
   - Uses YAML-based **cloud-config** files.
   - Example:

   ```
   #cloud-config
   timezone: Africa/Dar_es_Salaam
   hostname: test-system
   apt_update: true
   apt_upgrade: true
   packages:
     - nginx
   ```

   - Reduces manual setup for new VMs or containers.

9. Containers

1. **Container Concepts**

   - Isolated environment for an application.
   - Shares host OS kernel, thus lighter than full VMs.
   - Faster deployment and scaling, easy migration.

2. `cgroups` **(Control Groups)**

   - Linux kernel feature limiting resource usage (CPU, memory, IO).
   - Container engines (Docker, LXC, Kubernetes) use cgroups under the hood.

3. **Use Cases**

   - Microservices, ephemeral workloads, dev/test environments.

Workbook Exercises

1. **Compare VM Types**

   - Write down 3 differences between **fully virtualized** and **paravirtualized** VMs.
   - List examples of **Type-1** vs. **Type-2** hypervisors.

2. **Inspect a VM Definition (libvirt)**

   - On a KVM host, look at `/etc/libvirt/qemu/VM_NAME.xml`.
   - Identify the disk image file, CPU count, and memory assignment.

3. **Check Machine ID**

   - On a Linux VM, run:

   ```
   dbus-uuidgen --get
   ```

   - If cloned, try regenerating the machine ID.
   - Discuss why identical IDs can cause conflicts.

4. `cloud-init` **Basics**
    - Create a small `cloud-config` file to set a hostname and install a package.
    - Discuss how it might be used in a real deployment scenario.
5. **Container vs. VM**
    - Compare resource usage for a container vs. a full VM (e.g., Docker container vs. KVM instance).
    - List potential advantages of containers in your environment.

Summary

- **Linux** supports various virtualization technologies (KVM, Xen, VirtualBox), each with different performance and integration trade-offs.
- Paravirtualization leverages special drivers for higher performance than fully virtualized guests.
- **D-Bus machine ID** and **SSH keys** must be unique for each cloned VM or template-based deployment.
- `cloud-init` automates initial OS configuration in cloud or container environments.
- **Containers** share the host kernel, providing lighter, faster deployment compared to full VMs, and rely on `cgroups` for resource isolation.

Multiple-Choice Questions for 102.6

1. Which hypervisor is described as a Type-1 (bare-metal) hypervisor that does **not** rely on an underlying operating system?

   A) VirtualBox

   B) VMware Workstation

   C) KVM

   D) Xen

2. What is the main purpose of a **guest driver** in a paravirtualized environment?

   A) They hamper performance by adding extra overhead

   B) They replace the hypervisor entirely

   C) They help the guest OS interact with the hypervisor hardware more efficiently

   D) They prevent kernel modules from loading

3. Which of the following statements about disk images is **correct**?

   A) The raw image format is always smaller in physical size

   B) A 10 GB raw image file only uses 5 GB by default

   C) Copy-on-write images cannot support snapshots

   D) `qcow2` is a copy-on-write disk image format used by QEMU

4. Which statement accurately describes **containers**?

   A) Containers require a fully emulated BIOS and disk controllers

   B) Containers cannot be migrated from one host to another

   C) Containers are identical to fully virtualized machines

   D) Containers isolate applications while sharing the host's operating system kernel

5. Which of the following is an example of a **Type-2** hypervisor mentioned in the text?

   A) VirtualBox

   B) Xen

   C) KVM

   D) Docker

6. Which command can be used to ensure a system has a D-Bus machine ID or to generate one if missing?

   A) `systemctl machine-id`

   B) `uuidgen`

   C) `cloud-init -machine-id`

   D) `dbus-uuidgen -ensure`

7. Which best describes **cloud-init** as mentioned in the text?

   A) A virtualization environment used to create containers

   B) A network configuration tool for bridging

   C) A proprietary cloud computing platform

   D) A vendor-neutral utility for automatically configuring new cloud-based systems at first boot

8. What is the recommended procedure when cloning a Linux VM that needs a **unique** D-Bus machine ID?

   A) Reboot the system, and it will generate a new ID automatically

   B) No action is needed; the hypervisor handles ID generation

   C) Remove `/etc/machine-id` and generate a new one with `dbus-uuidgen`

   D) Request a new license from LPI

9. Which is **true** regarding copying SSH public keys with the `ssh-copy-id` command?

   A) `ssh-copy-id` can only be used on local machines, not remote servers

   B) `ssh-copy-id` places the public key into the `authorized_keys` file on the remote server

   C) The private key is automatically transferred to the remote server

   D) `ssh-copy-id` sets the public key file permission to 700

10. In a libvirt network configuration, which statement is correct regarding **bridging**?

    A) Bridging is never used by VMs

    B) The `default.xml` might define a bridge interface named `virbr0`

    C) NAT is never used with bridging

    D) The bridging device must have the same name as the hypervisor

11. Which of the following statements is **true** about NAT in the libvirt `default` network definition?

    A) NAT is never used in libvirt

    B) The default network uses NAT to forward packets to other networks

    C) NAT requires advanced bridging configuration

    D) NAT can only be used with a single VM

12. Which file typically stores a **symbolic link** to `/etc/machine-id`?

    A) `/usr/lib/dbus/machine-id`

    B) `/run/machine-id`

    C) `/var/lib/dbus/machine-id`

    D) `/etc/dbus/machine-id`

13. Which virtualization disk provisioning approach **only** grows in size as new data is written to the disk image?

    A) RAW

    B) Copy-on-write (COW)

    C) Partition-based allocation

    D) LVM-based thick provisioning

14. Which hypervisor is described in the text as both Type-1 **and** Type-2 because it integrates with the Linux kernel but also runs on a host OS?

    A) KVM

    B) VirtualBox

    C) Xen

    D) VMware ESXi

15. Which of the following are considered **IaaS computing elements** for cloud-based virtualization?

    A) Computing instances, block storage, and virtual networking

    B) Word processors, spreadsheets, and messaging apps

    C) Email, databases, and printers

    D) Standard Operating Procedures (SOPs)

16. When using `ssh-keygen` to generate an SSH key pair, which file extension typically indicates the **public** key file?

    A) `.priv`

    B) `.pub`

    C) `.asc`

    D) `.id`

17. What is the main advantage of paravirtualized drivers (guest drivers) over fully virtualized drivers?

    A) They are less secure

    B) They require specialized hardware that is not widely supported

    C) They typically offer better performance by allowing the guest OS to interact directly with the hypervisor

    D) They reduce memory usage by 70%

18. Which of the following statements about **container technology** is correct?

    A) It always requires a separate OS kernel per container

    B) It is always slower than a fully virtualized solution

    C) It allows applications to run in isolated environments while sharing the host kernel

    D) Containers cannot be migrated between hosts

19. Which command is used to **add** a public SSH key to the remote server's `authorized_keys` file automatically?

    A) `ssh-copy-id`

    B) `scp`

    C) `scp-pub`

    D) `sftp`

20. Which statement accurately describes **live migration** in virtualization?

    A) Live migration is the process of moving a running VM from one hypervisor to another without downtime

    B) Live migration means the guest OS must be halted first

    C) Live migration is only possible with container technology

    D) Live migration requires external storage with no snapshots

Fill-in-the-Blank Questions for 102.6

1. The software platform responsible for managing hardware resources for virtual machines is called the ___________.
2. When a virtual machine is aware that it is a VM and uses specialized drivers, it is referred to as a ___________ guest.
3. The ___________ file format (used by QEMU) supports copy-on-write functionality.
4. In a KVM setup, the XML configuration files for virtual machines are often located under ___________.
5. The ___________ command can generate a new D-Bus machine ID if one does not already exist.
6. A symbolic link for the machine ID is typically found at '/var/lib/dbus/machine-id', pointing back to ___________.
7. When a virtual machine is copied to act as a ___________, certain unique properties (like SSH keys or machine IDs) must be changed.
8. ___________ is a vendor-neutral utility used to automatically configure new cloud-based virtual machines at first boot.
9. An example of a **Type-2** hypervisor, mentioned in the text, that runs on top of an existing OS is ___________.
10. ___________ is a method that allows a virtual machine to be moved from one hypervisor to another with minimal or no downtime.

Chapter 3

Topic 103: GNU and Unix Commands

103.1 Work on the command line

Reference to LPI Objectives

- **LPIC-1 version 5.0, Exam 101, Objective 103.1**
- **Weight:** 4

Key Knowledge Areas

- Using single shell commands and one-line command sequences.
- Managing the shell environment: defining, referencing, and exporting variables.
- Using and editing command history.
- Invoking commands inside and outside of the PATH.

Important Commands, Files, and Concepts

- **bash** (shell)
- **echo**, **env**, **export**
- **pwd**, **set**, **unset**
- **type**, **which**
- **man**, **uname**
- **history**, **.bash_history**
- **Quoting** (single quotes, double quotes, backslash)

Lesson Overview

Mastering the command line is foundational for Linux administration. You'll frequently need to view or modify your environment, recall and repeat past commands, and handle special characters. Below are the essentials of working efficiently from the shell.

1. Basic System and Command Information

1. **Where Am I?**: `pwd`
   - Prints your current directory, e.g., `/home/user`.
   - Example:

   ```
   pwd
   # /home/frank
   ```

2. **System Information**: `uname -a`
   - Displays kernel name, version, architecture, and more.
   - Example:

```
uname -a
# Linux base 4.18.0-18-generic ...
```

3. **Manual Pages**: `man COMMAND`
   - Displays documentation for a specified command.
   - If unsure of the exact command name, use `apropos KEYWORD`.
4. **Command Identification**:
   - `type COMMAND`
     - Tells whether it's a shell builtin, a hashed command, or an external binary.
   - `which COMMAND`
     - Shows the absolute path (e.g., `/usr/bin/ls`).

2. Using Command History

1. **Listing Past Commands**
   - `history`
     - Shows a list of your previously executed commands.
   - Piping to `grep KEYWORD` can search through it:

   ```
   history | grep apt
   ```
2. **.bash_history**
   - Hidden file in your home directory storing commands.
   - Only updates when you exit a session, so the most recent commands may not appear until logout.
3. **Re-executing Commands**
   - **Up/Down Arrow** keys cycle through your history.
   - Press **Enter** to execute.
   - Saves time re-typing complex commands.

3. Environment Variables

1. **Listing Environment Variables**
   - `env` shows exported variables (visible to child processes).
   - `set` shows all variables and shell functions.
2. **Viewing Variable Values**
   - `echo $VARIABLE_NAME`
   - Example:

   ```
   echo $PATH
   ```

3. **Creating and Exporting Variables**
   - `VARIABLE=value` (local to the current shell).
   - `export VARIABLE` makes it inherited by child shells.
   - Example:

     ```
     myvar=hello
     export myvar
     ```

4. **Removing Variables**
   - `unset VARIABLE` deletes it from the current environment.

4. Quoting and Special Characters

1. **Why Quote?**
   - Spaces and certain symbols are interpreted by the shell.
   - Quoting ensures the literal interpretation of special characters/spaces.

2. **Methods**
   - **Double quotes** (`" "`): preserves most characters except $, `, \, and **!** in some cases.
   - **Single quotes** (`' '`): preserves all characters literally.
   - **Backslash** (\): escapes just the next character.

3. **Examples**
   - Creating a file with spaces:

     ```
     touch "my big file"
     ```

   - Removing it:

     ```
     rm 'my big file'
     ```

   - Escaping spaces:

     ```
     touch my\ big\ file
     ```

Workbook Exercises

1. **Check Your Current Directory**
   - Run `pwd` and verify the exact path to your home directory.
   - Create a file there using `touch <filename>`.

2. **Find Your Kernel Version**
   - Use `uname -a` and note the kernel version.
   - Check `man uname` to see other possible options.

3. **Explore Man Pages**
   - Run `man ls` and look for the `-l` option description.

- Use `apropos kernel` to see commands/man pages referencing "kernel."

4. **Practice Command History**
   - Execute 5–10 random commands (like `pwd`, `ls`, `echo test`).
   - Run `history` and filter with `grep ls`.
   - Press the **Up** arrow key to retrieve a previous command and re-run it.
5. **Experiment with Environment Variables**
   - Create a variable: `myvar="test123"`.
   - Echo it: `echo $myvar`.
   - Start a new shell with `bash`, check if `myvar` is available.
   - Go back, export `myvar`, start another shell, and see if it's now available.
   - Remove it with `unset myvar`.
6. **Creating Files with Special Characters**
   - Try `touch my big file` (observe the result).
   - Now properly create the file: `touch "my big file"`.
   - Remove it in three different ways (double quotes, single quotes, backslash-escaped).

Summary

- The `pwd` and `uname` commands help locate you and your system's details.
- `man`, `apropos`, `type`, and `which` help you find and understand commands.
- `history` and the `.bash_history` file let you recall and reuse previous commands.
- Environment variables (`PATH`, etc.) are easy to manage with `export`, `unset`, and `echo`.
- Quoting (single quotes, double quotes, or backslashes) is crucial when dealing with spaces or special characters.

Multiple-Choice Questions for 103.1

1. Which of the following commands quickly displays only the absolute pathname of an executable (without additional information)?

   A) type

   B) file

   C) whereis

   D) which

2. When using `bash`, which key sequence allows you to recall and edit previously typed commands?

   A) Ctrl+H and Ctrl+G

   B) The Up/Down arrow keys

   C) Left-clicking on the command line

   D) The Tab key

3. Which of the following commands can be used to remove an environment variable from the current shell session?

   A) unsetenv

   B) erase

   C) export –remove

   D) unset

4. Which `uname` option prints all available system information?

   A) `-r`

   B) `-o`

   C) `-a`

   D) `-v`

5. Which command will remove the file `my big file` if the file name has embedded spaces and you do NOT use quotes or backslashes correctly?

   A) `rm my_big_file`

   B) `rm my big file` (interpreted as removing three separate files)

   C) `rm 'my big file'`

   D) `rm "my big file"`

6. Which command outputs the environment variables that are exported and accessible to child processes?

   A) env

   B) set

   C) grep

   D) apropos

7. Which of the following statements is **true** regarding `type uname` showing "uname is hashed (/bin/uname)"?

   A) It means `uname` was used previously and is cached for faster lookups

B) It means the command is a shell builtin

C) It means the command no longer exists on disk

D) It means there is a conflict with the `uname` command location

8. By default, new local variables set in the Bash shell are only available:

A) In the current shell session

B) To all newly created shells and user sessions

C) After a reboot only

D) To all users on the system

9. When you type `set | grep myvar` and see a result, but `env | grep myvar` returns nothing, what does that tell you about `myvar`?

A) `myvar` is stored in `.bashrc`

B) `myvar` is a local shell variable (not exported)

C) `myvar` is inherited from a parent environment

D) `myvar` is actually a path variable

10. Which command is used to list all recent commands executed in the current user's shell session?

A) man

B) history

C) ls -a

D) more

11. The file `~/.bash_history` typically contains:

A) A script that runs every time you open your shell

B) A record of previously executed commands in Bash

C) User-defined functions

D) Environment variables that persist after logout

12. Which command is the quickest way to verify whether the directory `/usr/local/bin` is in your `$PATH`?

A) `echo $PATH`

B) `man path`

C) `apropos local`

D) `ls /usr/local/bin`

13. Which command is most suitable for searching through the names and descriptions of all installed man pages when you do **not** remember the exact command name you need?

A) more

B) info

C) apropos

D) tail

14. Which of these statements regarding `man` is **true**?

A) `man` files are stored in `/etc/bash_completion.d/`

B) `man` pages are often organized into separate sections

C) `man` only lists synonyms for commands; it does not provide usage

D) `man` must be run as `root` to view system documentation

15. If you type `myvar=hello` (with no spaces), what happens?

A) This sets an environment variable globally for all shells

B) A local shell variable `myvar` is created with value `hello`

C) The variable is appended to the path

D) You must run `env` to permanently store that variable

16. Which of the following methods will **not** preserve special characters in a filename?

A) Using double quotes

B) Using single quotes

C) Escaping them with a backslash (\)

D) Typing them as is, without quotes or escapes

17. If you run `exit` within a child shell, what happens?

A) You return to the parent shell

B) The system reboots

C) It logs you out completely

D) The variable `$PATH` is cleared

18. Which command's output is typically the largest and includes all variables and functions (both local and exported)?

A) env

B) type

C) apropos

D) set

19. Which of these commands is a **shell builtin** by default on most Linux systems using Bash?

A) uname

B) cp

C) which

D) kill

20. What does pressing the **Up Arrow** key multiple times in Bash do?

A) Automatically corrects the last typed command

B) Displays man pages for previously run commands

C) Logs the user out if pressed 3 times in succession

D) Cycles through the recent command history

103.2 Process text streams using filters

Reference to LPI Objectives

- **LPIC-1 v5, Exam 101, Objective 103.2**
- **Weight:** 2

Key Knowledge Areas

- Sending text files and output streams through standard text utility filters.
- Familiarity with GNU textutils (now part of GNU coreutils) and related commands (`sed`, `grep`, `head`, `tail`, etc.).

Important Commands and Utilities

- `bzcat`, `cat`, `cut`, `head`, `less`, `md5sum`
- `nl`, `od`, `paste`, `sed`, `sha256sum`, `sha512sum`
- `sort`, `split`, `tail`, `tr`, `uniq`, `wc`
- `xzcat`, `zcat`
- Redirection operators (>, ») and **pipes** (|).

1. Quick Review: Redirections and Pipes

1. **Standard Streams**
   - **stdin** (standard input): file descriptor 0 (keyboard by default).
   - **stdout** (standard output): file descriptor 1 (screen by default).
   - **stderr** (standard error): file descriptor 2 (screen by default).
2. **Redirections**
   - > → redirect stdout to a file (overwrite).
   - » → redirect stdout to a file (append).
   - < → redirect a file into stdin.
3. **Pipes** (|)
   - Direct output of one command as input to another.
   - Example:

```
cat file.txt | grep "pattern"
```

2. Basic Usage of `cat`

1. Concatenate Files: `cat file1 file2` → writes both files to stdout in sequence.
2. Standard Input: Just `cat` (with no arguments) reads from stdin (keyboard).
3. Copying Files: `cat source > destination`.
4. Appending: `echo "new line" » file.txt`.

3. Viewing Compressed Files

- `bzcat` → for `.bz2` compressed files.
- `xzcat` → for `.xz` compressed files.
- `zcat` → for `.gz` compressed files.
- Example:

```
gzip file.txt # produces file.txt.gz
zcat file.txt.gz
```

4. Searching Text

- `grep`:
  - Search for lines matching a pattern: `grep pattern file`.
  - Common options:
    * `-i` → ignore case.
    * `-v` → invert match (show lines *not* matching).
    * `-n` → show line numbers.
- Example:

```
grep -i "this" mytextfile
# matches "This" or "this"
```

5. Paging Through Large Output

1. `less`:
   - Interactive pager: scroll with arrow keys, search with `/pattern`.
   - Example:

   ```
   less /var/log/syslog
   ```

2. `head` and `tail`:
   - `head file` → first 10 lines.
   - `tail file` → last 10 lines.
   - `-n <count>` → changes how many lines are shown (e.g., `head -n 5`).
3. `nl` and `wc`:
   - `nl` → numbers each line of input.
   - `wc` → word count, line count, etc.
   - `wc -l` → line count only.

6. Editing Text Streams with `sed`

1. **Pattern Matching**
   - Print only lines matching a regex: `sed -n '/regex/p' file`.
   - Delete lines matching: `sed '/regex/d' file`.
2. **Find and Replace**
   - `sed 's/old/new/' file`.
   - In-place edit: `sed -i.backup 's/old/new/' file`.
3. Example:

```
sed -n /cat/p < ftu.txt # prints lines containing "cat"
sed /cat/d < ftu.txt # prints everything except lines containing "cat"
```

7. Ensuring Data Integrity with Checksums

1. **Checksum Tools**: `md5sum`, `sha256sum`, `sha512sum`.
2. **Generating a Hash**:

```
sha256sum ftu.txt > sha256.txt
```

3. **Verifying a Hash**:

```
sha256sum -c sha256.txt
# ftu.txt: OK
```

8. Looking Deeper with `od` (Octal Dump)

1. **Default**
   - `od file` → displays file contents in octal.
   - Often for debugging binary or unusual text files.
2. **Common Options**
   - `-x` → display as hexadecimal.
   - `-c` → display as characters (escaped for non-printable).
   - `-An` → suppress addresses/offset.
3. **Example**

```
    od -c file
    # shows hidden characters like \n
```

Workbook Exercises

1. **Basic Redirection**
   - Create `test.txt`, then run `cat > test.txt` and type some lines, press `Ctrl+C` to end.
   - Use `diff` or `cat` to confirm contents.
2. **Pipes**
   - `ls -l /etc | grep conf`
   - `cat /etc/passwd | wc -l` (count lines in `/etc/passwd`).
3. **Compressed File Viewing**
   - Create a large text file (`ls -R /usr > big.txt`).
   - Compress it with `gzip big.txt`.
   - Use `zcat big.txt.gz | head`.
4. **Searching & Paginating**
   - `grep "root" /etc/passwd`
   - `less /var/log/syslog` (scroll, search for "error" with `/error`).
5. **sed Basics**
   - `sed -n '/root/p' /etc/passwd` → lines containing "root."
   - `sed 's/bash/sh/' /etc/passwd | head` → replace "bash" with "sh," show first 10 lines.
6. **Checksum**
   - Run `sha256sum ftu.txt > check.txt`.
   - Modify `ftu.txt` and verify using `sha256sum -c check.txt` to observe the mismatch.
7. **Examining File Contents**
   - `od -c ftu.txt` → see hidden newline chars.
   - `od -x ftu.txt` → observe hex representation.

Summary

- Redirection and pipes let you chain commands and outputs.
- Powerful text filters include `grep`, `head`, `tail`, `less`, `nl`, `wc`, and `sed`.
- Checksum commands (`md5sum`, `sha256sum`, `sha512sum`) ensure data integrity.
- Use `od` to reveal hidden or binary data in files.
- Mastering these techniques streamlines text processing, a crucial skill for Linux administration.

Multiple-Choice Questions for 103.2

1. Which command reads from standard input if no file is specified and echoes the input to standard output?

   A) grep

   B) tail

   C) sed

   D) cat

2. Which redirection operator **creates** or **overwrites** a file with the output of a command?

   A) >

   B) »

   C) |

   D) &>

3. Which command can be used to display specific lines that match a pattern in a text file, **ignoring** case differences when the `-i` option is used?

   A) sort

   B) head

   C) grep

   D) md5sum

4. Which command is commonly used to paginate output, allowing you to scroll through text using the arrow keys?

   A) less

   B) cat

   C) nl

   D) tail

5. Which command shows only the last ten lines of a file by default?

   A) head

   B) tail

   C) cut

   D) split

6. Which command can be used to generate or check the cryptographic integrity of a file using **MD5** hashing?

   A) md5sum

   B) sha256sum

   C) sha512sum

   D) od

7. Which command is considered a **stream editor** that can filter and transform text, including find-and-replace operations?

   A) nl

B) wc

C) uniq

D) sed

8. Which command can be used to **display** a file in **octal** or **hexadecimal** representation (helpful for debugging)?

   A) less

   B) paste

   C) od

   D) zcat

9. If you want to **append** the output of a command to an existing file (without overwriting), which redirection operator should you use?

   A) >

   B) &>

   C) 2>

   D) »

10. Which command, by default, displays the **first 10 lines** of a file?

    A) tail

    B) wc

    C) head

    D) cut

11. Which command is used to **decompress and display** the content of a `.gz` file without explicitly creating an uncompressed file?

    A) gzip

    B) bzcat

    C) xzcat

    D) zcat

12. Which option with `gzip` activates **verbose** mode to show what is happening during compression?

    A) -v

    B) -n

    C) -c

    D) -q

13. Which command can **enumerate** lines of output (by placing a line number at the beginning of each line)?

    A) wc

    B) nl

    C) sed

    D) sha512sum

14. Which command can **merge lines** from multiple files side-by-side into columns?

A) sort

B) uniq

C) paste

D) split

15. Which command is commonly used to **sort** lines of text in **alphabetical** or **numerical** order?

    A) split

    B) sort

    C) tr

    D) nl

16. Which command is used to **transform** or **translate** characters from standard input (for example, converting uppercase to lowercase)?

    A) od

    B) cat

    C) tr

    D) head

17. Which tool can be used to verify file integrity using an **SHA-256** hash?

    A) sha256sum

    B) zcat

    C) md5sum

    D) tail

18. Which of these commands **decompresses** a file using the **bzip2** algorithm and sends its content to standard output?

    A) zcat

    B) bzcat

    C) sha512sum

    D) nl

19. Which of the following commands will **remove duplicate lines** from a sorted list?

    A) paste

    B) uniq

    C) od

    D) grep

20. Which command is used to **split** large files into smaller parts?

    A) tr

    B) md5sum

    C) nl

    D) split

103.3 Perform basic file management

Reference to LPI Objectives

- **LPIC-1 v5, Exam 101, Objective 103.3**
- **Weight:** 4

Key Knowledge Areas

- Copying, moving, and removing files/directories (individually and recursively).
- Using wildcards (file globbing) for matching patterns.
- Locating files using `find` (by type, size, time).
- Using `tar`, `cpio`, `dd` for archiving, copying, and backup tasks.

Important Commands and Utilities

- `cp`, `mv`, `ls`, `rm`, `rmdir`, `mkdir`, `touch`
- `find`
- `tar`, `cpio`, `dd`
- `gzip`, `gunzip`, `bzip2`, `bunzip2`
- `file` (to identify file type)
- Wildcards: `*`, `?`, `[ ]`

1. File Listing and Basic Navigation

1. `ls`

- `ls` lists contents of a directory.
- Common options:
  - `-l` – long listing (permissions, owner, size, date/time).
  - `-a` – include hidden files (dotfiles).
  - `-h` – human-readable sizes.
  - `-R` – list contents recursively.
- Example:

```
ls -lh /var/log
```

2. `touch`

- Creates empty files or updates file timestamps.
- Example:

```
touch myfile.txt
# creates an empty file if it doesn't exist
```

2. Creating and Removing Directories

1. `mkdir`

- Make new directories.
- `mkdir dir1` – creates `dir1`.
- `mkdir -p parents/children` – creates a nested directory path if it doesn't already exist.

2. `rmdir`

- Remove empty directories.
- Fails if directory is not empty.
- `rmdir -p parents/children` – removes nested directories if all are empty.

3. Copying, Moving, and Deleting Files

1. `cp` (Copy)

- `cp file1 dir2` – copy `file1` into `dir2`.
- `cp -r dir1 dir2` – copy directory `dir1` recursively into `dir2`.
- Useful options:
  - `-i` – prompt before overwrite.
  - `-f` – force overwrite.

2. `mv` (Move / Rename)

- `mv file1 dir2` – move `file1` into `dir2`.
- `mv oldname newname` – rename a file.
- Options:
  - `-i` – prompt before overwrite.
  - `-f` – force.

3. `rm` (Remove)

- `rm file1 file2` – remove multiple files.
- `rm -r dir1` – remove `dir1` and its contents recursively.
- `rm -i file1` – prompt before removal.
- `rm -f file1` – force removal (no prompt).
- **WARNING**: `rm -rf /` is very dangerous.

4. Wildcards (File Globbing)

- `*` (asterisk): matches zero or more characters.
  - Example: `ls *.txt` – lists all `.txt` files.
- `?` (question mark): matches exactly one character.
  - Example: `ls l?st.txt` – matches `last.txt`, `lest.txt`, `list.txt`.
- `[ ]` (brackets): matches any one character from the group/range inside.
  - Example: `ls file[1-3].txt` – matches `file1.txt`, `file2.txt`, `file3.txt`.

Wildcards can be combined:

```
ls [plf]?st*
# e.g., matches last.txt, lest.txt, list.txt, past.txt
```

5. Finding Files: `find`

```
find STARTING_PATH [OPTIONS] [EXPRESSION]
```

1. Search by Name

- `find . -name "myfile.txt"` – find `myfile.txt` in current directory.
- `find /home -iname "*.png"` – case-insensitive, all `.png` under `/home`.

2. Search by Type

- `-type f` (regular file), `-type d` (directory), `-type l` (symlink).
- Example: `find /var -type d -name "log"`.

3. Search by Size

- `-size +2G` (bigger than 2GB).
- `-size -20M` (smaller than 20MB).
- Suffixes: `b` (bytes), `k` (KB), `M` (MB), `G` (GB).

4. Search by Modification Time

- `-mtime N` →changed exactly `N` days ago.
- `-mtime +N` →changed more than `N` days ago.
- `-mtime -N` →changed less than `N` days ago.
- Example:

  ```
  find /etc -name "*.conf" -mtime -3
  # conf files changed less than 3 days ago
  ```

5. Act on Results (`-exec`)

- `-exec COMMAND {}` – run a command on each match.
- Example:

```
find . -name "*.bak" -exec rm {} \;
```

- Or use `-delete` to remove matches automatically:

```
find . -name "*.bak" -delete
```

6. Archiving and Compression

6.1 `tar`

1. **Creating an Archive**

```
tar -cvf archive.tar dir1 dir2
# -c: create, -v: verbose, -f: specify file
```

2. **Extracting**

```
tar -xvf archive.tar
# -x: extract
```

3. **Compression**

   - `-z` for gzip →`.tar.gz` or `.tgz`:

   ```
   tar -czvf archive.tar.gz dir1
   tar -xzvf archive.tar.gz
   ```

   - `-j` for bzip2 →`.tar.bz2`:

   ```
   tar -cjvf archive.tar.bz2 dir1
   tar -xjvf archive.tar.bz2
   ```

6.2 `gzip` / `bzip2`

- `gzip file` →creates `file.gz`.
- `gunzip file.gz` →uncompress.
- `bzip2 file` →creates `file.bz2`.
- `bunzip2 file.bz2` →uncompress.

6.3 `cpio`

- Create:

```
ls | cpio -o > archive.cpio
```

- Extract:

```
cpio -id < archive.cpio
```

6.4 dd

- General form: `dd if=INFILE of=OUTFILE [options]`.
- Example: copy a file:

```
dd if=oldfile of=newfile status=progress
```

- Convert text to uppercase:

```
dd if=oldfile of=newfile conv=ucase
```

- Disk backup (be cautious):

```
dd if=/dev/sda of=backup.dd bs=4096
```

Workbook Exercises

1. **Basic Operations**
    - Create a directory `testdir` with `mkdir testdir`.
    - Inside `testdir`, create files (`touch file1 file2`).
    - List them (`ls -l`), then copy them into a new directory `copydir`.
    - Rename one file in `copydir` to `file3`.
    - Remove `copydir` recursively.
2. **Globbing**
    - Create files: `fileA`, `fileB`, `fileX`, `file12`, `f_test`, etc.
    - Use wildcards to list or remove subsets (`ls f*`, `rm file?`, etc.).
3. **Finding Files**
    - Run:

    ```
    find . -name "*.sh"
    ```

    - Search by size (`+1M`, etc.).
    - Use `-exec echo {}` to print each match.
4. **Archiving & Compressing**
    - Create a tar archive of a test directory:

    ```
    tar -cvf myarchive.tar testdir
    ```

    - Compress it (`gzip myarchive.tar`) or do it in one step (`-z`):

    ```
    tar -czvf myarchive.tgz testdir
    ```

    - Extract into a new location:

    ```
    tar -xzvf myarchive.tgz -C /tmp
    ```

5. **Using `dd`**
    - Copy a file using `dd`, e.g.:

    ```
    dd if=testfile of=testfile_copy bs=1K status=progress
    ```

    - Verify both files with `diff` or `cmp`.

Summary

- `ls` shows file details; `mkdir`, `rmdir` create/remove directories.
- `cp`, `mv`, `rm` handle copying, moving, renaming, and deleting files/directories.
- Use `-r` for recursive operations on directories.
- File globbing (`*`, `?`, `[ ]`) simplifies specifying multiple files in commands.
- `find` locates files by name, type, time, or size, and can act on them using `-exec` or `-delete`.
- `tar`, `cpio`, `dd` provide archiving, backup, and data copying capabilities, optionally with compression (gzip, bzip2).

Multiple-Choice Questions for 103.3

1. Which command is used to list files in the current directory in a **human-readable** format (showing sizes like 4.0K, 2.1M, etc.)?

   A) ls -lh

   B) ls -lr

   C) ls -a

   D) ls -sd

2. Which option with the `rm` command **prompts** the user before removing the file?

   A) -r

   B) -f

   C) -i

   D) -d

3. In the context of wildcard usage, what does the `*` (asterisk) match?

   A) Zero or more occurrences of **any** character

   B) Exactly one character

   C) Only hidden files

   D) Only directories

4. Which command **removes** an **empty directory**?

   A) rm -r

   B) rmdir

   C) rm -rf

   D) cpio

5. Which option for the `cp` command is used to copy directories **recursively**?

   A) -v

   B) -l

   C) -u

   D) -r

6. In the output of `ls -l`, which character (in the first column) indicates a **directory**?

   A) -

   B) s

   C) c

   D) d

7. Which command can be used to **create an empty file**?

   A) touch

   B) newfile

   C) cp -0

   D) mv -n

8. What is the effect of running `mv oldname newname`?

   A) Copies `oldname` to `newname`

   B) Renames `oldname` to `newname`

   C) Asks for confirmation before removing `oldname`

   D) None of the above

9. Which command displays the **type** of a file (e.g., text, directory, etc.)?

   A) file

   B) type

   C) grep

   D) dd

10. Which command is used to **remove** a file **permanently** without prompts, ignoring nonexistent files?

    A) rm -f

    B) rm -i

    C) rm -r

    D) rm -rf

11. Which command will **list hidden files** in the current directory?

    A) ls -l

    B) ls -h

    C) ls -a

    D) ls -R

12. Which of the following wildcards matches **exactly one character**?

    A) *

    B) !

    C) ?

13. Which command can **create** a directory named `parents/children` along with the necessary **parent** directory if it doesn't exist?

    A) mkdir -r parents/children

    B) mkdir –create parents/children

    C) mkdir -p parents/children

    D) mkdir -m parents/children

14. What does the following command do?

    ```
    cp -r myfiles/ backups/
    ```

    A) Copies the entire `myfiles` directory, including its contents, to `backups`

    B) Moves all content from `backups` to `myfiles`

    C) Copies only hidden files from `myfiles` to `backups`

D) Backs up the entire filesystem

15. Which command is used to **move or rename** files in Linux?

    A) cp

    B) rename

    C) mv

    D) rm

16. When using the **mv** command, which option **prompts** you before overwriting an existing file?

    A) -f

    B) -n

    C) -r

    D) -i

17. In the context of the `find` command, what does `-type f` represent?

    A) Search for directories

    B) Search for regular files

    C) Search for symbolic links

    D) Search for special files

18. Which command **recursively removes** a directory and all its contents?

    A) rmdir -r

    B) rm -r

    C) rm

    D) rmdir

19. Which command can be used to **list** the content of a directory **together with its subdirectories**?

    A) ls -R

    B) ls -a

    C) ls -l

    D) ls -m

20. Which wildcard expression would match filenames that start with `l`, followed by any **single** character, and end with `st.txt`?

    A) l*[st].txt

    B) l?st.txt

    C) l??st?.txt

    D) l?st.[txt]

103.4 Use streams, pipes and redirects

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 103.4**
- **Weight:** 4

Key Knowledge Areas

- Redirecting standard input (stdin), standard output (stdout), and standard error (stderr).
- Piping output of one command into another command's input.
- Using output of one command as arguments to another command.
- Sending output to both stdout and a file.

Important Commands, Files, and Utilities

- `tee`, `xargs`
- Redirection operators (>, », <, «, «<, 2>, &>, etc.)
- Pipes (|)
- Command substitution (`‘command‘` or `$(command)`)

1. Standard File Descriptors

1. **stdin**: file descriptor **0** (normally keyboard input).
2. **stdout**: file descriptor **1** (normally terminal display).
3. **stderr**: file descriptor **2** (normally terminal display for errors).

2. Redirection

1. Output Redirection

- > : redirect stdout (file descriptor 1) to a file (overwrite).
- »: append stdout to a file.
- 2>: redirect stderr (file descriptor 2).
- &> or >&: redirect **both** stdout and stderr to a file.
- Example:

```
command > file.txt # overwrites file.txt
command >> file.txt # appends to file.txt
command 2> errors.txt # only stderr to errors.txt
command &> all_output.txt # stdout and stderr
```

2. Input Redirection

- < : read file content into stdin.
- Example:

```
command < file.txt
```

- Usually, commands can also specify a file directly (e.g., `cat file.txt`), but < can be used if needed.

3. Here Documents and Here Strings

1. Here Document («)

- Multi-line string as stdin to a command.
- Terminates on a line containing a marker (like `EOF`).

```
cat << EOF
line 1
line 2
EOF
```

- Everything up to the terminating word is fed to the command's stdin.

2. Here String (< < <)

- Single-line string to a command's stdin.
- Example:

```
wc -c <<< "hello world"
```

- If the string contains spaces, put it in quotes.

4. Pipes

1. Definition

- | connects stdout of one command to stdin of another.
- Multiple pipes can chain many commands.

2. Basic Example

- `cat file.txt | grep "pattern"`
- The `cat` output goes to grep's input.

3. Combining with Redirection

- `command1 2>&1 | command2` merges stderr into stdout first, so second command can read both.

4. tee

- Splits output so you can see on screen **and** write to a file.
- `command | tee file.txt`
  - Output goes to screen and also saved into `file.txt`.
- `-a` appends instead of overwriting.

5. Command Substitution

1. Syntax

- `` ̀command` `` (backticks) or `$(command)`
- Example (using `$()` recommended):

```
TODAY=$(date +%Y-%m-%d)
echo "Today's date is $TODAY"
```

- Used to store command output in variables or pass as arguments to other commands:

```
mkdir dir-$(date +%Y%m%d)
```

6. Using xargs

1. Purpose

- Takes a list (from stdin) and builds arguments for another command.
- Commonly used with `find` output or other multi-line data.

2. Basic Example

```
find . -name "*.txt" | xargs rm
```

- This removes all `.txt` files found.
- If filenames have spaces, use `-print0` in `find` and `-0` in `xargs`:

```
find . -name "*.txt" -print0 | xargs -0 rm
```

3. Options

- `-n 1`: run the specified command once per line.
- `-I {}`: placeholder to control where the arguments go in the command line.
- Example:

```
find . -name "*.jpg" | xargs -I {} mv {} /tmp
```

  - Moves each `.jpg` to `/tmp`.

Workbook Exercises

1. Redirecting Output

- Run `ls -l /nonexistentdir 2> error.txt`.
- Inspect `error.txt` to see stderr captured.

2. Piping

- Display line count of `/etc/passwd`:

```
cat /etc/passwd | wc -l
```

- Or shorter: `wc -l < /etc/passwd`.

3. Here Document

- Create a test file with multiple lines:

```
cat << EOF > mytest.txt
line1
line2
line3
EOF
```

- Check contents with `cat mytest.txt`.

4. tee

- Pipe a command's output to a file and terminal:

```
ls -l | tee listing.txt
```

5. Command Substitution

- Store today's date in a variable:

```
TODAY=$(date +%Y-%m-%d)
echo $TODAY
```

- Use it to create a directory, e.g. `mkdir backup-$TODAY`.

6. xargs

- Create some test files, e.g. `touch file1 file 2 "file space.txt"`.
- Use `find` and `xargs` with `-print0`/`-0` to remove them:

```
find . -name "file*" -print0 | xargs -0 rm
```

Summary

- **stdin** (0), **stdout** (1), and **stderr** (2) are standard I/O channels.
- **Redirects** (>, », 2>, <, «, etc.) move data between commands and files.
- **Pipes** (|) send one command's output to another command's input.
- **tee** duplicates data to both stdout and a file.
- **Command substitution** (`` `cmd` `` / `$(cmd)`) captures a command's output for variables or arguments.
- **xargs** transforms stdin lines into command arguments, often used with `find`.

Multiple-Choice Questions for 103.4

1. What character is used to create a pipeline in Linux?

   A) <

   B) »

   C) |

   D) &

2. In a pipeline, data flows from ___ to ___.

   A) Filesystem to memory

   B) Left to right

   C) Right to left

   D) Memory to filesystem

3. What command allows the output of a pipeline to be displayed on the screen and written to a file simultaneously?

   A) tee

   B) uniq

   C) grep

   D) wc

4. Which of the following is a valid method to redirect standard error to standard output in Bash?

   A) |

   B) 2>&1

   C) <&

   D) »

5. In Bash, the `$(...)` syntax is used for ___.

   A) Redirection

   B) Command substitution

   C) Background execution

   D) Piping

6. Which command is used to pass the output of one program as arguments to another program?

   A) uniq

   B) xargs

   C) grep

   D) wc

7. When using `find` with `xargs`, which option ensures correct handling of paths with spaces?

   A) -L

   B) -n 1

   C) -print0 and -0

   D) -exec

8. Which command is used to sort output numerically?
   A) tee
   B) uniq
   C) sort -n
   D) wc
9. The `xargs` option ___ specifies how many arguments to use per command execution.
   A) -L
   B) -I
   C) -n
   D) -0
10. In a pipeline, which program skips duplicate lines?
    A) wc
    B) tee
    C) uniq
    D) grep
11. Which of the following can capture only the standard output of a process in a pipeline?
    A) pipe (|)
    B) Redirect (>)
    C) uniq
    D) grep
12. Which `xargs` option allows substituting input values anywhere in the target command?
    A) -n
    B) -L
    C) -I
    D) -0
13. Command substitution in Bash allows you to ___.
    A) Send stderr to stdout
    B) Use command output as an argument
    C) Redirect stdin to a file
    D) Save stdout in a file
14. The `-exec` option in `find` ___.
    A) Runs a command for each search result
    B) Sorts the output of find
    C) Moves files to a directory
    D) Saves search results to a file
15. When using `xargs`, what option limits the number of lines used as arguments per execution?
    A) -print0

B) -L

C) -n

D) -exec

16. How can you ensure the `xargs` command processes paths with special characters?

A) Use -n 1

B) Use -print0 and -0

C) Use tee

D) Use uniq

17. Which of the following combines input redirection and piping?

A) `cat /proc/cpuinfo | wc`

B) `grep 'model name' </proc/cpuinfo | uniq`

C) `uniq | wc`

D) `tee`

18. What does the `-a` option in the `tee` command do?

A) Appends output to a file

B) Prevents overwriting a file

C) Redirects output to stderr

D) Filters duplicate lines

19. The `find` option `-mindepth 2` instructs the command to ___.

A) Process files only at a specified depth

B) Include hidden files in the results

C) Skip symbolic links in the search

D) Sort results by size

20. What is the purpose of using `xargs` with `-I` in a pipeline?

A) Handle special characters

B) Use the input value at a specific position

C) Limit arguments per command

D) Run a command for each result

103.5 Create, monitor and kill processes

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 103.5**
- **Weight:** 4

Key Knowledge Areas

- Running jobs in foreground and background
- Keeping processes running after logout (e.g., `nohup`)
- Monitoring active processes (e.g., `ps`, `top`, `free`, `uptime`, `watch`)
- Selecting/sorting processes (e.g., `ps` options, `top` interactions)
- Sending signals to processes (e.g., `kill`, `pkill`, `killall`)
- Using terminal multiplexers (`screen`, `tmux`)

Important Commands, Files, and Utilities

- **bg**, **fg**, **jobs**
- **kill**, **nohup**
- **ps**, **top**
- **free**, **uptime**, **watch**
- **pgrep**, **pkill**, **killall**
- **screen**, **tmux**

1. Foreground and Background Jobs

1. Foreground Execution

- By default, commands run in foreground and occupy the terminal until complete.
- Example: `sleep 60` (blocks the terminal for 60 seconds).

2. Suspending a Foreground Job

- Press **Ctrl** + **Z** to suspend (stop) a running job.

3. Background Execution

- Append & to run a process in the background immediately:

```
sleep 60 &
```

- Or after suspending, use `bg` to continue a stopped job in the background.

4. jobs

- Lists active jobs associated with the current shell.
- Each job has a **job ID** (e.g., `[1]`).

5. fg

- Brings a background (or stopped) job to the foreground.
- Example: `fg %1` → bring job #1 forward.

6. Terminating a Job

- Use `kill %1` to send a signal (default SIGTERM) to job #1.
- If a job does not respond, try `kill -9 %job_number` (SIGKILL).

2. Detaching Jobs: `nohup`

1. `nohup`

- Runs a command immune to **SIGHUP** (hangup signal).
- The process continues after you log out.
- Example:

```
nohup ping localhost > ping.log 2>&1 &
```

- Output by default goes to `nohup.out` if not redirected.

2. Killing a nohup Process

- Identify the PID (e.g., `ps aux | grep ping` or `pgrep ping`) and then `kill <PID>`.

3. Monitoring Processes

1. watch

- Periodically runs a command (default every 2 seconds).
- Example:

```
watch uptime
watch -n 5 free
```

- Press **Ctrl** + **C** to exit.

2. free

- Displays memory usage (RAM, swap).
- Use `-m` or `-h` for human-readable format.

3. uptime

- Shows system up time, number of users, load averages.

4. ps

- Snapshots current processes.
- Common options:
  - `ps aux` (show all processes in BSD style)
  - `ps -ef` (show all processes in UNIX style)
  - `ps -U <user>` or `ps -user <user>` (filter by user)
- Fields: **PID**, **USER**, **%CPU**, **%MEM**, **VSZ** (virtual size), **RSS** (resident set size), **TTY**, **STAT**, **TIME**, **COMMAND**.

5. top

- Dynamic, real-time view of processes.
- Interactive keys (some important ones):
  - **M** (sort by memory usage)
  - **P** (sort by CPU usage)
  - **k** (kill a process by PID)
  - **r** (renice a process by PID)
  - **q** (quit)

4. Sending Signals to Processes

1. kill

- Sends a signal to a process by PID or job spec.
- Default is `-TERM (SIGTERM, signal #15)`.
- `-9` (SIGKILL) forcibly kills.
- Example:

```
kill -TERM 1234
kill -9 1234
kill %1 # kills job #1
```

2. killall

- Kills all processes by command name.
- Example: `killall firefox`.

3. pkill

- Kills processes by name or matching pattern (use `pgrep` first to see matches).
- Example: `pkill -9 sleep`.

4. Signal Reference

- `kill -l` → list all signals.
- Common signals:
  - SIGHUP (1), SIGINT (2), SIGQUIT (3), SIGTERM (15), SIGKILL (9), SIGSTOP (19), SIGCONT (18).

5. Terminal Multiplexers: `screen` and `tmux`

5.1 `screen`

1. Basic Usage

- Start: `screen` (creates a new session).
- Detach a session: **Ctrl** + **A** then **D**.
- Reattach: `screen -r [session_id]` (or `screen -ls` to list sessions).
- Kill a session: `screen -S <session_id> -X quit`.

2. Windows

- Each window is like a separate terminal.
- Create new window: **Ctrl** + **A** then **C**.
- Move between windows: **Ctrl** + **A** then **N** (next) or **P** (previous).
- Rename a window: **Ctrl** + **A** then **Shift** + **A** (enter new name).

3. Regions (Split Screen)

- Horizontal split: **Ctrl** + **A** then **S**.
- Vertical split: **Ctrl** + **A** then `|`.
- Move between splits: **Ctrl** + **A** then **Tab**.

4. Copy Mode

- Enter copy mode: **Ctrl** + **A** then `[`.
- Use arrow keys to navigate, press **Space** to start selection, move, and **Space** again to end selection.
- Paste in a window: **Ctrl** + **A** then `]`.

5.2 `tmux`

1. Basic Usage

- Start: `tmux` (creates a new session).
- Detach: **Ctrl** + **B** then **D**.
- List sessions: `tmux ls`.
- Attach: `tmux attach -t <session_name>`.
- Kill session: `tmux kill-session -t <name>`.

2. Windows

- New window: **Ctrl** + **B**, **C**.
- Switch windows: **Ctrl** + **B**, **N** (next) or **P** (previous).
- Rename: **Ctrl** + **B**, `,`.

3. Panes (Splits)

- Horizontal split: **Ctrl** + **B**, `"`.
- Vertical split: **Ctrl** + **B**, `%`.
- Switch panes: **Ctrl** + **B**, **Arrow keys**.
- Kill pane: **Ctrl** + **B**, `x`.

4. Copy Mode

- Enter: **Ctrl** + **B**, `[`.
- Move around, press **Space** to start selection, arrow, press **Space** again to finish.
- Paste: **Ctrl** + **B**, `]`.

Workbook Exercises

1. **Foreground/Background Jobs**
   - Run `sleep 60` in the foreground, suspend with **Ctrl** + **Z**.
   - Check `jobs`.
   - Send it to background with `bg`.
   - Bring it back to foreground with `fg`.
   - Kill it with `kill %1`.
2. **nohup**
   - Run `nohup ping localhost > ping.log 2>&1 &`.
   - Log out and back in.
   - Check the process with `ps` or `pgrep`.
   - Kill it with `kill <PID>`.
3. **Monitoring System Load**
   - Run `watch uptime` or `watch -n 5 free`.
   - Exit with **Ctrl** + **C**.
   - Compare to `top` output.
4. **Killing Processes**
   - Start a dummy process like `sleep 9999 &`.
   - Find the PID via `ps`, `pgrep sleep`.
   - Kill it with `kill <PID>`.
   - Confirm it's gone (`ps` or `jobs`).

5. **Using `top`**
    - Invoke `top`.
    - Press **k** to kill a process (e.g., search a small CPU process).
    - Sort by memory usage with **M**, or CPU usage with **P**.
    - Quit with **q**.
6. `screen`
    - Start a session (`screen`).
    - Create multiple windows (**Ctrl** + **A**, **C**).
    - Detach (**Ctrl** + **A**, **D**) and reattach (`screen -r`).
    - Kill a session (`screen -S <session_id> -X quit`).
7. `tmux`
    - Start a session (`tmux`).
    - Create new windows (**Ctrl** + **B**, **C**).
    - Split horizontally (**Ctrl** + **B**, `"`) or vertically (**Ctrl** + **B**, `%`).
    - Detach (**Ctrl** + **B**, **D**) and reattach (`tmux a`).
    - Kill with `tmux kill-session -t <name>`.

Summary

- Background jobs (`bg`) and foreground (`fg`) are managed via **jobs**.
- `nohup` allows processes to keep running after logout.
- Process monitoring uses commands like `ps`, `top`, `free`, `uptime`, `watch`.
- Signals are sent with `kill` (by PID/job), `killall`, `pkill`, each defaulting to **SIGTERM** if unspecified.
- Terminal multiplexers `screen` and `tmux` provide multi-window, multi-pane sessions that can detach and persist in the background, invaluable for system administration over remote connections.

Multiple-Choice Questions for 103.5

1. What command is used to run a process in the background directly while starting it?

   A) fg

   B) &

   C) nohup

   D) jobs

2. Which command is used to send a job to the foreground?

   A) fg

   B) bg

   C) kill

   D) nohup

3. Which signal is sent by default if no specific signal is mentioned while using the `kill` command?

   A) SIGSTOP

   B) SIGHUP

   C) SIGTERM

   D) SIGKILL

4. What does the `-n` option in the `watch` command do?

   A) Changes the update interval

   B) Displays the process ID

   C) Lists only running jobs

   D) Lists only stopped jobs

5. What key is pressed in `top` to sort the output by memory usage?

   A) T

   B) N

   C) R

   D) M

6. What is the purpose of the `nohup` command?

   A) To bring a job to the foreground

   B) To monitor active processes

   C) To detach a job so it continues running after logout

   D) To kill a specific process by PID

7. Which command is used to kill all instances of a specific process by name?

   A) killall

   B) pkill

   C) kill

   D) nohup

8. What is the role of the `jobs` command in Linux?
   - A) To display all running processes
   - B) To list active jobs started by the current shell
   - C) To terminate jobs
   - D) To monitor memory usage
9. What does the `free` command display?
   - A) Load averages
   - B) Memory usage statistics
   - C) Running jobs
   - D) Network statistics
10. Which option in the `ps` command shows all processes regardless of terminal attachment?
    - A) -u
    - B) x
    - C) a
    - D) aux
11. What is a key feature of terminal multiplexers like GNU Screen and tmux?
    - A) They replace terminal emulators entirely.
    - B) They are used exclusively for SSH connections.
    - C) They allow multiple terminal sessions within a single terminal window.
    - D) They are required to run graphical applications.
12. Which command in tmux splits the current window into vertical panes?
    - A) Ctrl + b - "
    - B) Ctrl + b - q
    - C) Ctrl + b - %
    - D) Ctrl + b - |
13. What is the default command prefix for GNU Screen?
    - A) Ctrl + a
    - B) Ctrl + b
    - C) Ctrl + c
    - D) Ctrl + d
14. In tmux, which command creates a new session with a specific name?
    - A) tmux new -n NAME
    - B) tmux create NAME
    - C) tmux new -s NAME
    - D) tmux start -n NAME
15. Which of the following commands terminates the current pane in tmux?
    - A) Ctrl + b - &

B) Ctrl + b - X

C) Ctrl + b - q

D) Ctrl + b - x

16. How do you reattach to a detached session in tmux?

    A) tmux attach-session

    B) tmux attach

    C) tmux -r

    D) tmux reconnect

17. What is the main difference between screen "regions" and tmux "panes"?

    A) Panes are more customizable than regions.

    B) Panes are independent pseudo-terminals, while regions are not.

    C) Panes are temporary, but regions persist between sessions.

    D) Panes require a session to be detached to function.

18. In GNU Screen, which command prefix is used to rename a window?

    A) Ctrl + a - r

    B) Ctrl + a - A

    C) Ctrl + a - n

    D) Ctrl + a - "

19. Which tmux key combination detaches a session?

    A) Ctrl + b - a

    B) Ctrl + b - d

    C) Ctrl + b - t

    D) Ctrl + b - m

20. Which configuration file can be used to customize tmux on a per-user basis?

    A) /etc/tmux.conf

    B) /.tmux.conf

    C) /usr/share/tmux/tmux.conf

    D) /.tmux.config

103.6 Modify process execution priorities

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 103.6**
- **Weight:** 2

Key Knowledge Areas

- Knowing the default priority (niceness) of newly created processes.
- Running a program with a higher or lower priority than default.
- Changing the priority of running processes.

Important Terms and Utilities

- **nice**
- **ps**
- **renice**
- **top**

1. Understanding Linux Process Scheduling

- **Multi-tasking (multi-processing) Systems:** Multiple processes "share" CPU time.
- **Preemptive Scheduling:** The kernel can forcibly switch the CPU from one process to another (even if the current process never calls an I/O or system call).
- **Normal vs. Real-Time Scheduling:** Real-time processes have higher priority than normal processes. We generally only adjust normal processes using "nice" values.

Static Priorities & Niceness

- **Static Priorities:**
  - Range `0-99` for real-time processes.
  - Range `100-139` for normal processes (kernel internal representation).
- **Nice Values:**
  - Range `-20` (highest priority, "least nice") to `19` (lowest priority, "most nice").
  - Default nice value is `0` for normal processes.
  - Only `root` can set negative niceness.

2. Viewing Process Priorities

1. ps

- Example: `ps -el` or `ps -Al`
  - **PRI:** process priority (internal to kernel, typically 80 = normal + offset).
  - **NI:** nice value (`-20` to `19`).

2. top

- Interactive system monitor, displays **PR** (priority) and **NI** (nice).
- Priority shown in `top` for normal processes is `PR - 100`, typically ranging `0-39`.
- Press **r** in `top` to renice a process.

3. Running a Program with Custom Priority

nice

- **Syntax:**

```
nice -n <nice_value> command
```

- Example:

```
nice -n 10 tar czf backup.tar.gz /home
```

  - If `-n` is omitted, default niceness becomes `10`.
- A negative nice value (e.g., `-5`) raises priority (requires root privileges).

4. Changing Priority of Running Processes

renice

- **Syntax:**

```
renice <new_nice_value> [options] <target>
```

- Common options:
  - `-p <PID>`: target by process ID.
  - `-u <username>`: target all processes by user.
  - `-g <groupname>`: target all processes by group.
- Example (by PID):

```
renice +5 -p 1234
```

- Example (by user):

```
renice +10 -u alice
```

- Alternatively in `top`, press **r** to renice a running process.

Exercises

1. **Check Default Niceness**

   - Open a terminal and run:

   ```
   sleep 300 &
   ps -el | grep sleep
   ```

   - Note **NI** value (should be `0`).

2. **Start a Process with Custom Niceness**

   - Example:

   ```
   nice -n 5 gzip largefile
   ps -el | grep gzip
   ```

   - Observe **NI** column in `ps`.

3. **Renice a Running Process**

   - Find a CPU-bound process (e.g., `md5sum /dev/zero`), get PID via `pgrep md5sum`.
   - Increase niceness:

   ```
   renice +10 -p <PID>
   ```

   - Check new niceness in `top` or `ps`.

4. **Interactive with top**

   - Run `top`.
   - Press **r**, enter the PID, then a new nice value.
   - Verify changes in the NI column.

Summary

- `nice` sets a process's niceness at launch, default is `0`, range `-20` to `19`.
- `renice` changes niceness of an already running process.
- `top` and `ps` display priorities; niceness is shown in **NI** columns.
- Higher priority → lower niceness (negative). Lower priority → higher niceness (positive).
- Mastering process priorities optimizes CPU usage for critical or resource-intensive tasks.

Multiple-Choice Questions for 103.6

1. What is the default nice value assigned to a process in Linux?

   A) -20

   B) 0

   C) 10

   D) 19

2. How are real-time processes prioritized compared to normal processes in Linux?

   A) They are ignored.

   B) They have the same priority as normal processes.

   C) They have higher priority than normal processes.

   D) They have lower priority than normal processes.

3. What command is used to check the static priority of a process?

   A) top

   B) renice

   C) ps

   D) grep

4. In the ps command output, what column represents the priority?

   A) ADDR

   B) CMD

   C) PRI

   D) NI

5. Which of the following is a valid nice value range?

   A) -30 to 30

   B) -20 to 20

   C) 0 to 100

   D) -20 to 19

6. What is the default nice value assigned by the nice command if no option is specified?

   A) -20

   B) 0

   C) 19

   D) 10

7. How can the nice value of an already running process be changed?

   A) By using the nice command.

   B) By restarting the process.

   C) By using the kill command.

   D) By using the renice command.

8. What happens when a process with a lower nice value is introduced?

   A) It runs with a higher priority.

   B) It waits for other processes to complete.

   C) It gets executed last.

   D) Its priority remains unchanged.

9. Which scheduling priority range is assigned to normal processes?

   A) 0 to 99

   B) 0 to 120

   C) 100 to 139

   D) -20 to 19

10. What command can display dynamic priority changes continuously?

    A) ps

    B) top

    C) grep

    D) nice

11. Which key is used in the top command to modify a process's priority?

    A) r

    B) n

    C) p

    D) q

12. What does a nice value of -10 imply?

    A) Higher priority than default.

    B) Default priority.

    C) Lower priority than default.

    D) The process cannot run.

13. What is the function of the -p option in the renice command?

    A) Modify priority of all processes.

    B) Modify priority of all processes by a user.

    C) Specify a PID for priority modification.

    D) View current priorities.

14. What priority level corresponds to a real-time process with a static priority of 5?

    A) 135

    B) 5

    C) 40

    D) -5

15. Which process attribute dictates execution priority in a normal scheduling policy?

    A) Static priority

B) Nice value

C) System calls

D) File descriptors

16. What command shows the static priority of the systemd process?

A) ps -e

B) top

C) grep p̂rio /proc/1/sched

D) nice -n 0

17. In the ps command, which column displays the nice value?

A) CMD

B) PRI

C) NI

D) SZ

18. How does renice modify priorities for a user's processes?

A) -n

B) -p

C) -u

D) -g

19. What term describes Linux's ability to switch processes without waiting for system calls?

A) Nice value

B) Priority inversion

C) Preemption

D) Static scheduling

20. What does a nice value of 19 imply?

A) High priority.

B) Default priority.

C) Low priority.

D) Real-time scheduling.

Fill-in-the-Blank Questions for 103.6

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

103.7 Search text files using regular expressions

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 103.7**
- **Weight:** 3

Key Knowledge Areas

- Creating simple regular expressions with various notational elements.
- Understanding differences between basic (BRE) and extended (ERE) regular expressions.
- Recognizing special characters, character classes, quantifiers, and anchors.
- Using regular expressions to search file systems or file content.
- Deleting, changing, and substituting text with regex tools.

Important Commands and Utilities

- **grep**, **egrep** (or `grep -E`), **fgrep** (or `grep -F`)
- **sed**
- `regex(7)` (man page)
- `find` with `-regex`, `-iregex`

1. Regular Expression Basics

1. Atoms

- **Ordinary characters** match themselves (e.g., `b` matches "b" literally).
- **. (dot)** matches any single character.
- `^` anchors to the start of a line.
- `$` anchors to the end of a line.
- **Brackets [ ]** define a *bracket expression* (single atom that matches any character inside).
  - Example: `[0-9]` matches any digit from 0 to 9.
  - A caret `^` inside the bracket means negation: `[^ a-z]` matches non-lowercase letters.

2. Character Classes (in bracket expressions)

- `[[:digit:]]` for digits, `[[:alpha:]]` for letters, `[[:alnum:]]` for alphanumeric, etc.
- Only valid **inside** bracket expressions like `[[:digit:]]`.

3. Anchors

- `^` (start of line), `$` (end of line).
- Example: `^ abc$` matches lines that are *exactly* `abc`.

4. Basic (BRE) vs. Extended (ERE)

- **BRE:** `+`, `?`, and `{}` usually need backslashes (`\+`, `\?`, `\{i,j}`) for special meaning.
- **ERE:** `+`, `?`, `{}` directly recognized as quantifiers without backslashes.
- Tools:
  - `grep` uses BRE by default.
  - `egrep` (or `grep -E`) uses ERE.

5. Quantifiers

- `*`: zero or more occurrences of preceding atom.
- `+`: one or more occurrences (ERE or `\+` in BRE).
- `?`: zero or one occurrence (ERE or `\?` in BRE).
- `{i}`, `{i,}`, `{i,j}`: exact or range bounds (with backslash in BRE).

6. Branches and Backreferences

- **Branching:** `|` (ERE) or `\|` (BRE) for "OR" logic.
- **Backreferences:** `( )` group subexpressions (ERE) or `\( \)` (BRE), use `\1`, `\2` to refer to matched groups.

2. Searching with `grep` / `egrep` / `fgrep`

1. `grep`

- Default uses basic RE.
- Common options:
  - `-i` ignore case
  - `-v` invert match
  - `-n` show line numbers
  - `-r` recursive
  - `-w` match whole words
  - `-c` count matches
  - `-H` show filename
- Example:

```
grep "^error" /var/log/syslog
```

2. `egrep` (or `grep -E`)

- Uses extended RE.
- Example:

```
egrep "colou?r|color" file.txt
```

  - Matches `colour` or `color`.

3. `fgrep` (or `grep -F`)

- Interprets pattern literally (no special chars).
- Example:

```
fgrep "[a-z]*" file.txt
```

  – Finds lines containing `[a-z]*` literally.

4. Examples

- `grep -i "warning" /var/log/messages` → case-insensitive match "warning."
- `grep -c "^root" /etc/passwd` → count lines starting with "root."
- `grep -H -r "error" /var/log/` → recursively find "error" in `/var/log/`, show filenames.

3. Substitutions and Text Editing with `sed`

1. Basics

- Non-interactive stream editor, applies commands to each line.
- Syntax:

```
sed [options] -e COMMAND file
# or sed [options] -f SCRIPT file
```

2. Common Sed Commands

- `d`: delete the matching line(s).
- `c <text>`: replace the line with `<text>`.
- `s/FIND/REPLACE/[flags]`: substitute `FIND` with `REPLACE`.
  – `g` flag: replace all occurrences in that line (global).
  – Example:

```
sed 's/foo/bar/g' file.txt
```

- `p`: print the matching line.
- `a <text>`: append text after the matching line.
- `r <file>`: read content of `<file>` after matching line.
- Lines or ranges can be specified by line number or regex.
  – Example: `1d` deletes line 1, `2,5d` deletes lines 2 to 5, `/regex/d` deletes lines matching `regex`.

3. Combining

- Multiple commands can be separated by `;`:

```
sed -e '1d; /test/d' file.txt
```

  – Deletes line 1, then any line containing "test."

4. Examples Using find + regex

1. **find -regex**
    - Basic RE by default.
    - `-regextype posix-extended` or `-regextype egrep` for extended RE.
    - Example:

    ```
    find /usr/share/fonts -regextype egrep -regex ".*\.(ttf|otf|woff)$"
    ```

2. **find -iregex (case-insensitive)**
    - Example:

    ```
    find /home/user -iregex ".*\.pdf"
    ```

5. Combining Tools for Advanced Searches

- **Chaining:**
    - Pipe outputs through `grep`, `sed`, `uniq`, etc.
- **Example: Searching logs, removing duplicates**

```
lastb -n 10 -d -a --time-format notime \
    | grep -v '[0-9]$' \
    | sed 's/.* //' \
    | sort | uniq
```

- Explanation:
    - `lastb` shows bad login attempts, `-a` puts host at the end of the line, `-n 10` shows last 10.
    - `grep -v '[0-9]$'` filters out lines ending in digits (assuming those are raw IPs).
    - `sed 's/.* //'` keeps only the substring after the last space (the hostname).
    - Then `sort | uniq` eliminates duplicates.

Exercises

1. **Basic vs. Extended**
    - Create a text file with lines containing special chars (`+`, `?`, `()`, etc.).
    - Try matching them with `grep "..."` (basic RE) and `egrep "..."` (extended).
2. **Anchors and Dot**
    - Use `grep "Ĥello" file.txt` to find lines starting with "Hello."
    - `grep "world$" file.txt` to find lines ending with "world."
    - `grep "w.rld" file.txt` (dot matches any char).
3. **Bracket Expressions**
    - `grep "[[:digit:]]" file.txt` → lines with digits.
    - `grep "[A-Z]" file.txt` → uppercase letters.
4. **sed Substitution**

- `sed 's/foo/bar/g' file.txt` → replace "foo" with "bar."
- `sed '2,4d' file.txt` → delete lines 2 to 4.
- `/old/d` → delete lines containing "old."

5. **Combining**

   - `grep -i "error" /var/log/syslog | sed -n '/warning/p'`
     - Finds lines matching "error" and then prints only lines also having "warning."

6. **find with -regex**

   - `find / -regextype posix-extended -regex ".*\.conf"` → all `.conf` files.

Summary

- **Regular expressions** define a flexible method for searching (and sometimes replacing) text patterns.
- Basic RE often requires backslashes for special operators like `+`, `?`, `{}`, while extended RE does not.
- Core features include **atoms**, **character classes**, **anchors**, **quantifiers**.
- **Tools:**
  - `grep/egrep/fgrep` → quick searches in lines.
  - `sed` → stream editing and substitution.
  - `find` with `-regex` → RE-based filtering of file paths.
- Combining these tools (piping, advanced regex) enables powerful text processing and filtering workflows.

103.8 Basic file editing

Reference to LPI Objectives:

- **LPIC-1 v5, Exam 101, Objective 103.8**
- **Weight:** 3

Key Knowledge Areas

- Navigating a document using **vi** (or **vim**)
- Understanding vi **modes** and using them effectively
- Inserting, editing, deleting, copying, and finding text in vi
- Awareness of alternative editors (Emacs, nano, etc.)
- Setting or changing the **standard editor** (EDITOR or VISUAL environment variable)

Important Terms and Utilities

- **vi** (or **vim**)
- Search commands (`/`, `?`)
- Navigation keys (`h`, `j`, `k`, `l`)
- Insert modes (`i`, `o`, `a`)
- Deletion/copying (`d`, `p`, `y`, `dd`, `yy`)
- Exiting and saving (`ZZ`, `:w!`, `:q!`)
- **EDITOR** environment variable

1. Introducing vi

vi is a modal text editor commonly pre-installed on Unix-like systems. Two key modes to remember:

1. **Normal (Command) Mode**
   - vi **starts** in this mode.
   - Keystrokes perform navigation or editing commands.
   - Press `i` (insert) or `a` (append) to switch to Insert mode.
2. **Insert Mode**
   - Text typed appears in the document.
   - Press `Esc` to return to Normal mode.

Starting vi

```
vi <filename>
```

- `vi +<line> <file>` → jump to a line on opening.
- `vim` is a more feature-rich version, fully backward-compatible with vi.

2. Basic vi Navigation and Commands

In **Normal mode**:

Key	**Action**
`h, j, k, l`	Left, down, up, right (move cursor)
`0, $`	Move to start/end of current line
`G`	Go to end of file (use `#G` to go to line #)
`/pattern`	Forward search for `pattern` (press `n` to find next)
`?pattern`	Backward search (press `N` for next backward match)
`u`	Undo last action
`Ctrl+R`	Redo
`:`	Enter **Colon Command mode** (e.g., `:w`, `:q!`)

Basic Insert Commands

Key	**Action**
`i`	Insert at cursor
`a`	Append after cursor
`o`	Open new line below cursor and enter insert mode
`O`	Open new line above cursor
`Esc`	Leave insert mode, back to normal mode

3. Editing Text in vi

Deleting and Changing

Key	**Action**
`x`	Delete character under cursor
`dw`	Delete word (from cursor to next word boundary)
`dd`	Delete (cut) entire current line
`d$`	Delete from cursor to end of line
`cc`	Change entire line (delete and start insert mode)
`c3w`	Change next 3 words (example of “count + command”)

Copying and Pasting

Key	**Action**
`yy`	Yank (copy) current line
`y$`	Yank from cursor to end of line
`p`	Paste after cursor or below line
`P`	Paste before cursor or above line

Saving and Exiting

Key(s)	Action
`:w`	Write (save) changes
`:q`	Quit if no unsaved changes
`:wq` or `:x`	Save and quit
`:q!`	Quit **without** saving
`ZZ`	Save changes (if any) and quit (another shortcut)

4. Searching and Substituting in vi

- **Search:** Press / in normal mode, type a pattern, press **Enter**.
- **Backward search:** Press `?`, type a pattern, press **Enter**.
- **Substitute** (similar to `sed`):

```
:s/old/new/g
```

  - Replaces all occurrences of "old" with "new" in current line.
  - `:%s/old/new/g` → entire file.

5. Alternative Editors

1. **vim**
   - "vi improved," features syntax highlighting, multiple undo levels, plugins, etc.
2. **nano**
   - Simple, user-friendly. Insert mode by default, commands are **Ctrl** + key combos.
   - Common shortcuts:
     - **Ctrl** + **O** to save, **Ctrl** + **X** to exit, **Ctrl** + **K** to cut line, **Ctrl** + **U** to paste.
3. **Emacs**
   - Very powerful, integrates IDE features.
   - Commands use **Ctrl** + key or **Alt** + key combos.

6. Setting the Default Editor

- The shell checks `EDITOR` or `VISUAL` environment variables to decide which editor to use for tasks like `crontab -e` or `git commit`.
- Example:

```
export EDITOR=nano
```

- Add to ˜ /`.bash_profile` or similar to make persistent.

Exercises

1. **Basic vi**
   - Create a test file: `vi test.txt`.
   - Press `i`, enter some text. Press `Esc`.
   - Move around with `h`, `j`, `k`, `l`.
   - Delete a line with `dd`.
   - Save and exit: `:wq`.
2. **Navigation and Searching**
   - Use `:set number` inside vi to see line numbers.
   - Jump to line 5 with `5G`.
   - Search for "test" using `/test`.
   - Press `n` to find the next match.
3. **Substitutions**
   - In normal mode, type `:s/abc/xyz/g` to replace "abc" with "xyz" on the current line.
   - Apply to the entire file: `:%s/abc/xyz/g`.
4. **nano**
   - Run `nano test.txt`.
   - Type some lines, press `Ctrl + K` to cut a line, `Ctrl + U` to paste.
   - Save with `Ctrl + O`, exit with `Ctrl + X`.
5. **Default Editor**
   - `export EDITOR=vim`
   - Then `crontab -e` should open in vim.
   - Make it permanent in ˜/`.bashrc` or ˜/`.bash_profile`.

Summary

- **vi** (or vim) is a modal text editor: Normal mode (commands) and Insert mode.
- Basic movement: `h`, `j`, `k`, `l`; insertion: `i`, `a`, `o`; deleting: `d`, `x`; copying: `y`; pasting: `p`.
- Searching with `/pattern`; substituting with `:s/old/new/g`.
- Alternative editors: **nano**, **Emacs**, **vim** with advanced features.
- Shell environment variable `EDITOR` (or `VISUAL`) sets the default text editor for tasks like `crontab -e`.

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

Fill-in-the-Blank Questions (101.2)

1. BIOS, UEFI
2. 440, MBR
3. EFI System Partition (ESP)
4. init
5. Kernel parameters
6. regenerate
7. kernel ring buffer
8. POST (Power-On Self-Test)
9. inittab
10. daemon

101.3 Change Runlevels / Boot Targets and Shutdown or Reboot System

Multiple-Choice Questions (101.3)

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

Fill-in-the-Blank Questions (101.3)

1. /etc/inittab
2. telinit 1 or telinit s
3. telinit q
4. /etc/init.d/
5. unit
6.
   - multi-user.target
   - graphical.target
   - (Any valid systemd target name is acceptable here.)
7. isolate
8. systemd
9. wall
10. shutdown

Topic 102: Linux Installation and Package Management

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

102.1 Design Hard Disk Layout

Multiple-Choice Questions (102.1)

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

Fill-in-the-Blank Questions (102.1)

1. /home
2. /boot
3. partition
4. EFI System (or EFI System Partition / ESP)
5. mkswap
6. /var
7. mounting
8. Volume Group (VG)
9. swap file
10. LVM (Logical Volume Management)

102.2 Install a Boot Manager

Multiple-Choice Questions (102.2)

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

Fill-in-the-Blank Questions (102.2)

1. boot loader
2. Master Boot Record (MBR)
3. vmlinuz
4. initrd.img (or initial RAM disk)
5. System.map
6. /etc/default/grub
7. update-grub (or grub-mkconfig)
8. /boot
9. /boot/grub/menu.lst
10. chainloading

102.3 Manage Shared Libraries

Multiple-Choice Questions (102.3)

Fill-in-the-Blank Questions (102.3)

1. ldd
2. LD_LIBRARY_PATH
3. /etc/ld.so.conf.d
4. static
5. LD_LIBRARY_PATH
6. soname
7. ldconfig
8. sudo ldconfig
9. shared libraries
10. .a

102.4 Use Debian Package Management

Multiple-Choice Questions (102.4)

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

Fill-in-the-Blank Questions (102.4)

1. selections
2. sources
3. -f
4. information
5. cache
6. files
7. #
8. cache
9. clean
10. purge

102.5 Use RPM and YUM Package Management

Multiple-Choice Questions (102.5)

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

Fill-in-the-Blank Questions (102.5)

1. -e
2. apt
3. search, se
4. remove
5. /etc/yum.conf
6. all
7. check-update
8. info
9. dnf
10. -ql

102.6 Linux as a virtualization guest

Multiple-Choice Questions (102.6)

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

Fill-in-the-Blank Questions (102.6)

1. hypervisor
2. paravirtualized
3. qcow2
4. /etc/libvirt/qemu
5. dbus-uuidgen –ensure
6. /etc/machine-id
7. template
8. cloud-init
9. VirtualBox
10. Live migration

Topic 103: GNU and Unix Commands

103.1 Work on the command line

Multiple-Choice Questions (103.1)

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

Fill-in-the-Blank Questions (103.1)

103.2 Process text streams using filters

Multiple-Choice Questions (103.2)

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

Fill-in-the-Blank Questions (103.2)

103.3 Perform basic file management

Multiple-Choice Questions (103.3)

1. A
2. C
3. A
4. B
5. D
6. D
7. A

8. B
9. A
10. A
11. C
12. D
13. C
14. A
15. C
16. D
17. B
18. B
19. A
20. B

Fill-in-the-Blank Questions (103.3)

103.4 Use streams, pipes and redirects

Multiple-Choice Questions (103.4)

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

17. B
18. A
19. C
20. B

Fill-in-the-Blank Questions (103.4)

103.5 Create, monitor and kill processes

Multiple-Choice Questions (103.5)

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

Fill-in-the-Blank Questions (103.5)

103.6 Modify process execution priorities

Multiple-Choice Questions (103.6)

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

Fill-in-the-Blank Questions (103.5)

Topic 104: Devices, Linux Filesystems, Filesystem Hierarchy Standard

Topic 105: Shells and Shell Scripting

Topic 106: User Interfaces and Desktops

Topic 107: Administrative Tasks

Topic 108: Essential System Services

Topic 109: Networking Fundamentals

Topic 110: Security