# DUMPSARENA

# LPIC-1 Exam 101, Part 1 of 2, version 5.0

## LPI 101-500

**Version Demo**

**Total Demo Questions: 37**

**Total Premium Questions: 373**

**Buy Premium PDF**

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

# Topic Break Down

| Topic | No. of Questions |
|---|---|
| Topic 1, System Architecture | 68 |
| Topic 2, Linux Installation and Package Management | 37 |
| Topic 3, GNU and Unix Commands | 182 |
| Topic 4, Devices, Linux Filesystems, Filesystem Hierarchy Standard | 86 |
| Total | 373 |

QUESTION NO: 1

Creating a hard link to an ordinary file returns an error. What could be the reason for this?

**A.** The source file is hidden.

**B.** The source file is read-only.

**C.** The source file is a shell script.

**D.** The source file is already a hard link.

**E.** The source and the target are on different filesystems.

**ANSWER: E**

**Explanation:**

The source and the target being on different filesystems is a valid reason that creating a hard link fails. A hard link is not a separate copy of a file; it is an additional directory entry that points to the same inode as the existing file. Inodes are allocated and managed by an individual filesystem, so a directory entry on one filesystem cannot point to an inode located on another filesystem. Consequently, the kernel rejects an attempt to create a hard link across filesystem boundaries, commonly reporting an “Invalid cross-device link” error (EXDEV). This applies even when both filesystem mount points appear within the same directory tree, such as directories below `/home` and `/mnt`; their underlying filesystem identity is what matters. If a link across filesystems is required, a symbolic link can be used instead, because a symbolic link stores a pathname rather than another reference to the original inode. The `ln` utility documentation notes that hard links are restricted to the same filesystem, and the Linux `link(2)` system-call documentation identifies EXDEV for links attempted across mounted filesystems. See GNU Coreutils: ln invocation and Linux link(2) manual page.

**QUESTION NO: 2**

What is the process ID number of the init process on a SysV init based system?

**A.** -1

**B.** 0

**C.** 1

**D.** It is different with each reboot.

**E.** It is set to the current run level.

**ANSWER: C**

**Explanation:**

On a SysV init-based Linux system, `init` has process ID `1`. The kernel starts this userspace process after completing its own initialization. Traditionally, the kernel executes `/sbin/init`, which reads `/etc/inittab` and then starts the services and processes appropriate for the configured runlevel. Because it is the first userspace process, it is assigned PID 1 and becomes the ancestor of nearly all other userspace processes.

PID 1 has a special role beyond its startup sequence. When a process loses its parent, the kernel reparents that orphaned process to PID 1 (or, on modern systems, potentially to a configured child subreaper). Init is also responsible for reaping terminated child processes so that they do not remain as zombies. These responsibilities are why PID 1 is fundamental to normal system operation. The `proc` filesystem documents that process information is exposed in directories named by PID, so `/proc/1` represents init. See the Linux init(1) manual page and the proc(5) manual page for details.

**QUESTION NO: 3**

Which of the following shell redirections will write standard output and standard error output to a file named filename?

**A.** 2>&1 >filename

**B.** >filename 2>&1

**C.** 1>&2>filename

**D.** >>filename

**E.** 1&2>filename

**ANSWER: B**

**Explanation:**

`>filename 2>&1` writes both standard output and standard error to `filename`. The first redirection, `>filename`, is shorthand for `1>filename`: it opens `filename` for writing and connects file descriptor 1, standard output, to that file. Unless append syntax is used, the file is created if necessary or truncated before output is written. The second redirection, `2>&1`, duplicates the current destination of file descriptor 1 onto file descriptor 2. Because standard output has already been redirected to `filename`, standard error is consequently directed to the same open file. Redirections are processed in the order written, which is essential to this result. This form is commonly used after a command, for example `command >filename 2>&1`, when both normal command output and diagnostic messages should be collected in one file. The Bash manual documents descriptor duplication and the left-to-right processing of redirections; POSIX also specifies that redirections are evaluated from beginning to end. See the GNU Bash Redirections documentation and the POSIX Shell Command Language.

**QUESTION NO: 4**

Which of the following examples for Bash file globbing matches a file named root-can-do-this.txt when used in the directory holding that file? (Choose three correct answers.)

**A.** root*can?do-this.{txt,odt}

**B.** r[oOoO]t-can-do*.txt

**C.** {root,user,admin}-can-??-this.txt

**D.** root*can*do??this.txt

**E.** root***{can,may}-do-this.[tT][xX][tT]

**ANSWER: A B C E**

**Explanation:**

There are four matching patterns, so the instruction to choose three is inaccurate. In Bash pathname expansion, `*` matches any string, including an empty string; `?` matches exactly one character; bracket expressions such as `[oOoO]` match one listed character; and brace expansion produces each listed alternative before pathname expansion. Therefore, `root*can?do-this.{txt,odt}` expands to a pattern that matches the filename, with the wildcard portions matching the hyphens. `r[oOoO]t-can-do*.txt` matches because the bracket expression accepts the `o` in `root` and the final star covers `-this`. `{root,user,admin}-can-??-this.txt` produces a `root` variant in which the two question marks match `do`. Finally, consecutive stars retain the same matching behavior as a star, so `root***{can,may}-do-this.[tT][xX][tT]` matches the lowercase filename through its `can` brace alternative and case-inclusive extension character classes. See the GNU Bash Pattern Matching documentation and the GNU Bash Brace Expansion documentation.

**QUESTION NO: 5 - (SIMULATION)**

Which program updates the database that is used by the locate command?

**ANSWER: See the explanation for the answer**

**Explanation:**

The program is `updatedb`.

`updatedb` creates or refreshes the filename database searched by `locate`. It is commonly run automatically by a scheduled job.

## QUESTION NO: 6

The system configuration file named is commonly used to set the default runlevel. (Please provide the file name with full path information)

**A.** /etc/inittab

**ANSWER: A**

**Explanation:**

`/etc/inittab` is the traditional System V init configuration file used to define system initialization behavior, including the default runlevel. In an inittab-based system, the entry `id:N:initdefault:` specifies the runlevel that init enters after booting, where `N` is the configured runlevel number. For example, `id:3:initdefault:` sets the normal multiuser text-mode runlevel as the default. This file can also contain directives controlling terminal login processes, actions to take at particular runlevels, and responses to events such as Ctrl+Alt+Delete.

The wording “commonly used” is important in the LPIC-1 version 5.0 context: traditional SysVinit systems use `/etc/inittab`, while many modern distributions using systemd instead select the boot target through a symbolic link such as `/etc/systemd/system/default.target`. The requested default-runlevel configuration file, however, is specifically the classic SysVinit file `/etc/inittab`. The LPI objectives include recognizing SysVinit configuration and the function of the `initdefault` entry. See the LPI Exam 101 objectives and the inittab(5) manual page.

## QUESTION NO: 7

What does the command mount --bind do?

**A.** It makes the contents of one directory available in another directory

**B.** It mounts all available filesystems to the current directory

**C.** It mounts all user mountable filesystems to the user’s home directory

**D.** It mounts all file systems listed in /etc/fstab which have the option userbind set

**E.** It permanently mounts a regular file to a directory

**ANSWER: A**

**Explanation:**

The command `mount --bind` creates an additional mount point for an existing directory tree. Therefore, “It makes the contents of one directory available in another directory” is correct. Rather than mounting a new filesystem from a device, network share, or disk image, a bind mount exposes the same already-mounted filesystem location at a second path. For example, `mount --bind /srv/data /var/www/data` makes the directory tree rooted at `/srv/data` accessible through `/var/www/data` as well. Both paths refer to the same underlying files and directories, so changes made through either path are visible through the other.

Bind mounts are useful when an application, service, container, or chroot environment needs access to a directory at a particular path without copying its contents. The bind operation applies to the specified directory and, by default, does not automatically create separate bind mounts for nested mount points; the recursive form, `mount --rbind`, is used when that behavior is needed. Bind mounts are initially created with the same mount options as the original mount, though mount attributes can subsequently be adjusted where supported. The util-linux `mount(8)` documentation describes bind mounting as attaching a filesystem or part of a filesystem to another place in the file hierarchy. See the mount(8) manual page and the Linux kernel mount namespace documentation.

## QUESTION NO: 8

Which file should be edited to select the network locations from which Debian installation package files are loaded?

**A.** /etc/dpkg/dpkg.cfg

**B.** /etc/apt/apt.conf

**C.** /etc/apt/apt.conf.d

**D.** /etc/apt/sources.list

**E.** /etc/dpkg/dselect.cfg

## ANSWER: D

**Explanation:**

`/etc/apt/sources.list` is the traditional APT configuration file that defines the package archives from which Debian obtains package metadata and installation packages. Each repository entry specifies a source location, typically using a network URI such as `http://` or `https://`, along with the Debian suite and repository component(s), for example `main`. When APT commands such as `apt update` run, APT reads these source definitions to determine where to download package indexes; when software is installed or upgraded, it uses the same configured locations to retrieve the required package files.

The file may contain entries for official Debian mirrors, security repositories, update repositories, or an internally managed package mirror. Administrators can therefore select or change the network sources used by the system by editing `/etc/apt/sources.list` and then refreshing the package indexes with `apt update`. Modern Debian systems can also store additional source definitions as files in `/etc/apt/sources.list.d/`, but the named file `/etc/apt/sources.list` is the correct answer for the standard primary source-list configuration. Debian documents the format and purpose of source-list entries in the sources.list manual page and provides repository configuration guidance in the Debian Reference.

## QUESTION NO: 9

Which of the following shell commands makes the already defined variable TEST visible to new child processes? (Choose two.)

**A.** visible TEST

**B.** declare +x TEST

**C.** declare –x TEST

**D.** export TEST

**E.** export –v TEST

## ANSWER: D

**Explanation:**

`export TEST` is the command that marks the existing shell variable `TEST` for export. Exporting a variable adds it to the environment inherited by subsequently started child processes, such as commands, scripts, or subshells launched from the current shell. The variable remains available in the current shell as well; exporting changes its inheritance behavior rather than creating a separate variable. This is the standard shell mechanism for passing configuration values, paths, locale settings, and similar environment data to programs that the shell starts later. The POSIX specification defines `export` as giving the named variable the export attribute so that it is placed in the environment of subsequently executed commands. Bash documents the same behavior and notes that a variable with the export attribute is automatically exported to child processes. See the POSIX export utility specification and the GNU Bash built-in command documentation.

## QUESTION NO: 10

Which of the following vi commands deletes two lines, the current and the following line?

**A.** d2

**B.** 2d

**C.** 2dd

**D.** dd2

**E.** de12

## ANSWER: C

**Explanation:**

`2dd` deletes two complete lines beginning with the current line: the line containing the cursor and the immediately following line. In vi's command mode, `dd` is the linewise delete command, and a numeric count placed before it specifies how many lines the command affects. Therefore, the count `2` makes the line-delete operation act on two consecutive lines. The deleted text is also placed into a register, so it can normally be restored with a put command such as `p` if needed. This count-plus-command pattern is a core vi editing convention and is useful for applying many editing operations efficiently without repeating individual keystrokes. Vim, which documents and preserves standard vi-style command behavior, describes `dd` as deleting the current line and supports a preceding count to delete that number of lines. See the Vim help for deleting lines with dd and the Vim documentation index.

## QUESTION NO: 11

Which of the following are filesystems which can be used on Linux root partitions? (Choose two.)

**A.** NTFS

**B.** ext3

**C.** XFS

**D.** VFAT

**E.** swap

## ANSWER: B C

**Explanation:**

**ext3** and **XFS** can both be used for a Linux root filesystem. A root filesystem contains the operating system's directory hierarchy, beginning at `/`, so it must be a filesystem that the Linux kernel can mount during the boot process and that supports normal Unix filesystem features such as directories, ownership, permissions, symbolic links, and device files. ext3 is the traditional ext-family filesystem with journaling support and has long been supported as a root filesystem. XFS is a mature, high-performance journaling filesystem supported by the Linux kernel and is commonly used as the root filesystem by Linux distributions and installations. The boot environment must make the needed filesystem driver available when

mounting the root filesystem; in typical distribution kernels, ext3 and XFS support is built in or included in the initramfs. The Linux filesystems(5) manual page describes filesystem types supported by Linux, while the kernel's XFS documentation covers XFS as a native Linux filesystem. Therefore, the two filesystem types suitable for Linux root partitions here are ext3 and XFS.

## QUESTION NO: 12

What information can the lspci command display about the system hardware? (Choose THREE correct answers.)

**A.** Device IRQ settings

**B.** PCI bus speed

**C.** System battery type

**D.** Device vendor identification

**E.** Ethernet MAC address

## ANSWER: A B D

**Explanation:**

The `lspci` utility lists devices attached to PCI buses and, with verbose output such as `lspci -v` or `lspci -vv`, reports details obtained from PCI configuration space and related kernel information. **Device IRQ settings** can be shown for PCI devices, typically as an IRQ number or interrupt-routing information in verbose output. This helps an administrator identify the interrupt assigned to a device and inspect how the kernel exposes that device's resources.

**PCI bus speed** can also be displayed where the device and PCIe capabilities provide link information. Verbose PCIe output includes link capabilities and current link status, such as negotiated speed and width. For example, PCIe links may be reported with transfer rates such as 2.5, 5, 8, or more GT/s.

**Device vendor identification** is a fundamental part of `lspci` output. It identifies each PCI function by vendor and device, using the PCI ID database when available to translate numeric identifiers into readable manufacturer and product names. Numeric IDs can be explicitly displayed with options such as `-n` or `-nn`. The lspci manual page documents its verbose and numeric output modes, and the PCI ID Repository provides the vendor and device ID database used for name resolution.

## QUESTION NO: 13

To prevent users from being able to fill up the / partition, the directory should be on a separate partition if possible because it is world writeable.

**A.** /tmp

## ANSWER: A

**Explanation:**

`/tmp` is the correct directory. Under the Filesystem Hierarchy Standard, `/tmp` is intended for temporary files created by programs and users. It is normally configured as world-writable so that unprivileged users and applications can create temporary working files. Because temporary data can grow rapidly or be deliberately created in large quantities, placing `/tmp` on its own filesystem helps prevent that usage from exhausting the space available on the root filesystem.

Keeping adequate free space on `/` is important for normal system operation: system services need to write logs, maintain state, create lock files, and perform package-management operations. A dedicated `/tmp` filesystem confines temporary-file consumption to the capacity assigned to it. Administrators can also apply suitable mount options, such as `nodev`, `nosuid`, and, where operationally appropriate, `noexec`, to reduce exposure from temporary files. The standard sticky-bit permission

convention on `/tmp` permits shared file creation while protecting users from deleting or renaming temporary files owned by other users. See the Filesystem Hierarchy Standard entry for /tmp and the mount manual page.

## QUESTION NO: 14

Which statements about Bash output redirection are correct? (Choose TWO correct answers.)

**A.** > normally truncates an existing output file.

**B.** >> appends standard output to a file.

**C.** > redirects standard error by default.

**D.** >> reads standard input from a file.

**E.** > can only redirect output to terminal devices.

## ANSWER: A B

**Explanation:**

The operator > redirects standard output to a file, creating the file if necessary and truncating it if it already exists. The operator >> also redirects standard output to a file, but appends output to the end rather than truncating existing content.

Standard error is file descriptor 2, so it is not redirected by plain > unless explicitly requested. Bash redirection syntax is documented in the GNU Bash Reference Manual.

## QUESTION NO: 15

Which of the following are valid stream redirection operators within Bash? (Choose two.)

**A.** <

**B.** #>

**C.** %>

**D.** >>>

**E.** 2>&1

## ANSWER: A E

**Explanation:**

The valid Bash redirection operators are << and `2>&1`. The << operator starts a here-document: Bash reads subsequent input lines up to a specified delimiter and supplies that text as standard input to a command. For example, `cat <<EOF` causes `cat` to receive the following lines until it encounters `EOF`. This is a shell input-redirection construct and is frequently used in scripts to provide multi-line command input.

The `2>&1` operator redirects file descriptor 2, standard error, to the current destination of file descriptor 1, standard output. It is commonly combined with an output redirect, such as `command >output.log 2>&1`, to place both normal output and diagnostic messages in the same file. The ordering is significant because the standard-error descriptor is duplicated to wherever standard output points at that moment. Bash documents redirections as operations that alter the input and output file descriptors associated with a command. See the GNU Bash manual: Redirections and the GNU Bash manual: Here Documents.

## QUESTION NO: 16

Which command will uninstall a package but leave its configuration files in case the package is re-installed?

**A.** dpkg –s pkgname

**B.** dpkg –L pkgname

**C.** dpkg –P pkgname

**D.** dpkg –r pkgname

**ANSWER: D**

**Explanation:**

`dpkg -r pkgname` is the correct command because `-r`, also written as `--remove`, removes the installed package while retaining its dpkg-managed configuration files (conffiles). This is useful when software is being temporarily uninstalled or replaced but its local configuration should be preserved for a later reinstall. If the same package is installed again, dpkg can retain and reuse those saved configuration files, avoiding the need to recreate site-specific settings such as service configuration, package defaults customized by an administrator, or other conffiles tracked by the package system.

In dpkg terminology, removal and purging are distinct package states and operations. A removed package can still have configuration files recorded on the system, which allows package-management tools and administrators to distinguish a normal uninstall from a full cleanup. The dpkg manual documents `--remove` as removing an installed package while leaving configuration files in place; it also notes that this behavior differs from purging, which is intended for complete removal. See the official dpkg manual at dpkg(1) and the Debian reference discussion of package management at The Debian Reference.

## QUESTION NO: 17

Which of the following commands brings a system running SysV init into a state in which it is safe to perform maintenance tasks? (Choose TWO correct answers.)

**A.** shutdown -R 1 now

**B.** shutdown -single now

**C.** init 1

**D.** telinit 1

**E.** runlevel 1

**ANSWER: C D**

**Explanation:**

In the SysV init model, runlevel 1 is the single-user or maintenance runlevel. Entering this runlevel stops normal multi-user services and provides a minimal environment intended for administrative work, such as repairing filesystems, changing critical configuration, or recovering a system. The commands `init 1` and `telinit 1` both request that the SysV init process switch to runlevel 1. `telinit` is traditionally the command-line interface used to communicate a runlevel change to the running init process; on SysV-style systems, `init` can also be invoked with a runlevel argument for this purpose. Because both commands transition the machine to the same single-user maintenance state, they are equivalent correct ways to prepare the system for maintenance. The system's current and previous runlevels can subsequently be inspected with tools such as `runlevel`, but the key operational action here is requesting runlevel 1. See the SysV init and runlevel documentation in the init(8) manual page and the telinit documentation.

## QUESTION NO: 18

What is the output of the following command?

echo "Hello World" | tr -d aieou

**A.** Hello World

**B.** eoo

**C.** Hll Wrld

**D.** eoo Hll Wrld

**ANSWER: C**

**Explanation:**

The output is `Hll Wrld`. The `echo "Hello World"` command writes the text `Hello World`, followed by a newline, to standard output. The pipe passes that text to `tr`. In `tr -d aieou`, the `-d` option means delete every character found in the supplied character set. Here, that set contains the lowercase vowels `a`, `i`, `e`, `o`, and `u`. Therefore, the lowercase `e` and both lowercase `o` characters are removed from `Hello World`. The uppercase `H` and `W` remain because `tr` character matching is case-sensitive unless a character class or additional characters are explicitly supplied. The space is not in the deletion set, so it is retained, and the newline emitted by `echo` is retained as well. Consequently, the displayed line is `Hll Wrld`. GNU coreutils documents `-d` as deleting characters in the specified set; the POSIX specification also defines `tr` as translating or deleting characters from standard input. See the GNU Coreutils tr documentation and the POSIX tr specification.

## QUESTION NO: 19

Which of the following commands determines the type of a file by using a definition database file which contains information about all common file types?

**A.** magic

**B.** type

**C.** file

**D.** pmagic

**E.** hash

**ANSWER: C**

**Explanation:**

The `file` command determines a file's type by inspecting its contents rather than relying solely on its filename or extension. It compares data found in the file, especially identifying byte sequences known as "magic numbers," against rules stored in a magic definition database. On typical Linux systems, this database is commonly available as `/usr/share/misc/magic`, `/usr/share/file/magic`, or in a compiled form such as `magic.mgc`; the precise location varies by distribution. For example, running `file /bin/ls` identifies the executable format, while `file image.png` can recognize PNG data even if the file has been renamed without its usual extension. The command can also examine text encodings, scripts with interpreter directives, archives, and many other standard formats. The `-m` option permits specifying an alternative magic file, demonstrating directly that `file` uses a definition database to perform this classification. This behavior is part of the standard command-line file identification facility documented in the file(1) manual page. The underlying magic-file syntax and database mechanism are further documented in the magic(5) manual page.

## QUESTION NO: 20

Which fields are present in a standard `/etc/fstab` entry? (Choose THREE correct answers.)

**A.** Filesystem specification

**B.** Mount point

**C.** Filesystem type

**D.** Kernel module name

**E.** Partition-table type

**ANSWER: A B C**

**Explanation:**

A standard `/etc/fstab` entry contains six whitespace-separated fields. These include the filesystem specification, the mount point, the filesystem type, mount options, the dump backup field, and the fsck order field. The filesystem specification can be a device path, UUID, label, or another supported identifier.

The mount point identifies where the filesystem should appear in the directory tree, while the filesystem-type field identifies the expected format, such as `ext4`, `xfs`, or `vfat`. See the fstab(5) manual page.

## QUESTION NO: 21

Which of the following commands reboots the system when using SysV init? (Choose TWO correct answers.)

**A.** shutdown -r now

**B.** shutdown -r "rebooting"

**C.** telinit 6

**D.** telinit 0

**E.** shutdown -k now "rebooting"

**ANSWER: A C**

**Explanation:**

`shutdown -r now` is a valid reboot command because the `-r` flag requests a reboot after shutdown, while `now` supplies the required shutdown time and causes the transition to begin immediately. Before rebooting, `shutdown` can notify logged-in users and perform the normal orderly shutdown sequence, including terminating services and synchronizing filesystems. The shutdown command's syntax and reboot action are documented in the shutdown(8) manual page.

`telinit 6` is also correct for a system using SysV init. The `telinit` command asks the init process to change to a specified runlevel. SysV runlevel `6` is conventionally the reboot runlevel; init executes the scripts configured for that runlevel and restarts the machine. This is the direct runlevel-based method, rather than invoking the shutdown utility. The relationship between `telinit`, init, and SysV runlevels is described in the init(8) manual page. These commands represent the standard immediate reboot mechanisms expected in a traditional SysV-init environment.

## QUESTION NO: 22 - (SIMULATION)

Which command will disable swapping on a device? (Specify ONLY the command without any path or parameters.)

Swapoff

/sbin/swapoff

**ANSWER: See the explanation for the answer**

**Explanation:**

`swapoff`

This is the command name used to disable swapping. The question specifically requests no path or parameters, so do not use `/sbin/swapoff`.

## QUESTION NO: 23

During a system boot cycle, what program is executed after the BIOS completes its tasks?

**A.** The bootloader

**B.** The inetd program

**C.** The init program

**D.** The kernel

## ANSWER: A

**Explanation:**

The bootloader is executed after the BIOS has completed its initial firmware responsibilities in a traditional BIOS-based boot sequence. The BIOS performs hardware initialization and the power-on self-test, identifies a bootable device according to its configured boot order, and reads the device's boot code into memory. It then transfers execution to that code, which is the first stage of the bootloader. The bootloader's purpose is to locate and load the selected operating system kernel, commonly also providing a menu for choosing among installed kernels or operating systems and passing kernel command-line parameters. For Linux systems, GRUB is a widely used bootloader and may use multiple stages to access filesystems and load the kernel image and initial RAM filesystem. This sequencing distinguishes the firmware's role of finding executable boot code from the bootloader's role of preparing and starting the operating system. The GNU GRUB manual describes GRUB as a boot loader and documents its role in loading operating systems: GNU GRUB Manual. A technical description of the BIOS boot environment and the handoff through boot sectors is also available in the Linux kernel x86 boot protocol documentation.

## QUESTION NO: 24

Which of the following commands display the number of bytes transmitted and received via the eth0 network interfaced (Choose TWO correct answers.)

**A.** route -v via eth0

**B.** ip stats show dev eth0

**C.** netstat -s -i eth0

**D.** ifconfig eth0

**E.** ip -s link show eth0

## ANSWER: D E

**Explanation:**

`ifconfig eth0` displays the configuration and operational counters for the specified interface. Its output includes received and transmitted traffic totals, conventionally shown in the `RX packets` and `TX packets` lines with accompanying `bytes` values. These byte counters are cumulative kernel interface statistics, allowing an administrator to see the amount of data received and sent through `eth0` since the counters were last reset, typically at boot or when the interface was recreated.

`ip -s link show eth0` is the modern iproute2 command for obtaining the same class of per-interface statistics. The `-s` flag requests statistics, and the output contains separate `RX:` and `TX:` sections. In each section, the first numeric field is the byte count, followed by packet and error-related counters. Specifying `eth0` limits the output to that interface, making this command especially suitable for checking its transmitted and received byte totals. The `ip-link` documentation describes -

`s` as printing statistics, while the legacy `ifconfig` manual documents its interface-status display. See ip-link(8) and ifconfig(8).

## QUESTION NO: 25

Which command displays the contents of the Kernel Ring Buffer on the command line? (Provide only the command name without any options or path information)

**A.** dmesg

## ANSWER: A

**Explanation:**

The correct command is `dmesg`. It reads and displays messages from the kernel ring buffer, a memory area in which the Linux kernel records messages produced during system startup and normal operation. These messages commonly include hardware detection results, device-driver initialization, kernel warnings, boot-time events, and other diagnostic information. Running `dmesg` with no options prints the available kernel-message buffer content to the command line, satisfying the requirement to provide only the command name. Access to some kernel messages may be restricted on hardened systems through the kernel's `dmesg_restrict` setting; however, this does not change the command used to display the buffer. The util-linux `dmesg` manual describes the command as being used to "print or control the kernel ring buffer," while the Linux kernel documentation identifies the ring buffer as the source of kernel log messages exposed to userspace. See the dmesg(1) manual page and the Linux kernel sysctl documentation.

## QUESTION NO: 26

What is true regarding UEFI firmware? (Choose two.)

**A.** It can read and interpret partition tables

**B.** It can use and read certain file systems

**C.** It stores its entire configuration on the /boot/ partition

**D.** It is stored in a special area within the GPT metadata

**E.** It is loaded from a fixed boot disk position

## ANSWER: A B

**Explanation:**

**It can read and interpret partition tables** is correct because UEFI boot services identify bootable devices and partitions before an operating system is running. UEFI defines partition discovery using GUID Partition Table (GPT) structures and recognizes the EFI System Partition (ESP), which is identified by its GPT partition-type GUID. This permits firmware boot managers to locate eligible boot partitions without relying on legacy master boot record boot code.

**It can use and read certain file systems** is also correct. UEFI firmware includes file-system support sufficient to access EFI boot applications and related files on the ESP. The UEFI specification requires the ESP to use a FAT-based file system, commonly FAT32 in current installations. Firmware can therefore load an EFI executable directly from a path such as `\EFI\BOOT\BOOTX64.EFI`. This file-oriented boot approach is a key UEFI capability and lets a system select boot loaders through firmware boot entries rather than loading a fixed disk sector. The firmware's defined handling of GPT partitions and the FAT-formatted ESP is described in the UEFI Specifications; practical ESP layout and boot-file conventions are also summarized by the Arch Linux EFI System Partition documentation.

## QUESTION NO: 27

Which world-writable directory should be placed on a separate partition in order to prevent users from being able to fill up the / filesystem? (Specify the full path to the directory.)

**A.** /tmp

**ANSWER: A**

**Explanation:**

The correct path is `/tmp`. This directory is intended for temporary files created by programs and users, and it is normally writable by all users. Because temporary data can grow rapidly, mounting `/tmp` on its own filesystem limits any space exhaustion to that filesystem rather than allowing it to consume the free space of the root filesystem, `/`. Keeping adequate free space on the root filesystem is important because core operating-system services, logging, package management, and administrative tasks may need to create files there.

A separate `/tmp` filesystem can be configured as a dedicated disk partition, logical volume, or a temporary in-memory filesystem such as `tmpfs`, depending on local requirements. Its size can then be controlled independently through filesystem sizing or mount settings. The directory should retain its usual world-writable access mode with the sticky bit, commonly represented as `1777`; this permits users to create temporary files while restricting removal or renaming of another user's files. The Filesystem Hierarchy Standard identifies `/tmp` as the location for temporary files, and the `file-hierarchy` manual documents its temporary-file purpose. See the Filesystem Hierarchy Standard: /tmp and file-hierarchy(7).

**QUESTION NO: 28**

Which of the following commands will print important system information such as the kernel version and machine hardware architecture?

**A.** sysinfo

**B.** uname -a

**C.** lspci

**D.** arch

**E.** info

**ANSWER: B**

**Explanation:**

`uname -a` is the appropriate command because it requests all of the system-identification fields provided by `uname`. Its output includes the kernel name, network node hostname, kernel release, kernel version, machine hardware name (architecture), processor type, hardware platform where available, and operating system. This makes it a standard, quick way to inspect core information about the running Linux system from a shell.

The `-a` option is important for the wording of this question: running `uname` without options normally displays only the kernel name, whereas `uname -a` includes both the kernel version-related fields and the machine architecture. For a more focused architecture-only result, administrators may use `uname -m`, but the all-fields form is the best match when several categories of important system information are requested together.

The GNU Coreutils documentation describes `uname` and its all-information option at GNU Coreutils: uname invocation. The system-call and command manual also documents the fields reported by `uname` at man7.org: uname(1).

**QUESTION NO: 29**

Which statements about running filesystem checks with `fsck` are correct? (Choose TWO correct answers.)

**A.** `fsck` can invoke a checker appropriate for the filesystem type.

**B.** A filesystem should normally be unmounted before repair.

**C.** `fsck` is used to create a new filesystem.

**D.** `fsck` only works with network filesystems.

**E.** Repairing an actively mounted filesystem is always safe.

**ANSWER: A B**

**Explanation:**

`fsck` is a front-end that invokes a filesystem-specific checking utility, such as `e2fsck` for ext-family filesystems. Filesystems should normally be unmounted before repair is performed because checking or modifying a mounted, actively changing filesystem can produce inconsistent results or cause damage.

The `fsck` command is commonly run during boot when a filesystem needs checking, or manually from maintenance mode after the target filesystem has been unmounted. See the fsck(8) manual page and the e2fsck(8) manual page.

## QUESTION NO: 30

Which of the following files exist in a standard GRUB 2 installation? (Choose two.)

**A.** /boot/grub/stages/stage0

**B.** /boot/grub/i386-pc/lvm.mod

**C.** /boot/grub/fstab

**D.** /boot/grub/grub.cfg

**E.** /boot/grub/linux/vmlinuz

**ANSWER: B D**

**Explanation:**

In a conventional BIOS-based GRUB 2 installation, `/boot/grub/grub.cfg` is the generated GRUB configuration file. It contains menu entries, boot parameters, and instructions used by GRUB when presenting its boot menu and loading an operating system. Administrators normally do not edit this generated file directly; distribution-specific tools such as `update-grub` or `grub-mkconfig` create it from configuration sources and scripts.

The `/boot/grub/i386-pc/` directory contains GRUB 2 modules for the i386-PC BIOS platform. `lvm.mod` is one such module and provides GRUB support for reading filesystems located on Logical Volume Manager logical volumes. These platform modules are installed with the GRUB components and can be loaded by GRUB as required during boot. The directory name is platform-specific: a UEFI installation instead uses a target-specific directory such as `x86_64-efi`, but `grub.cfg` remains the central configuration file. GNU GRUB's manual documents the configuration file and modular architecture at https://www.gnu.org/software/grub/manual/grub/grub.html#Configuration and https://www.gnu.org/software/grub/manual/grub/grub.html#Dynamic-module-loading.

## QUESTION NO: 31

What of the following statements are true regarding /dev/ when using udev? (Choose TWO correct answers.)

**A.** Entries for all possible devices get created on boot even if those devices are not connected.

**B.** Additional rules for udev can be created by adding them to /etc/udev/rules.d/.

**C.** When using udev, it is not possible to create block orcharacter devices in /dev/ using mknod.

**D.** The /dev/ directory is a filesystem of type tmpfs and is mounted by udev during system startup.

**E.** The content of /dev/ is stored in /etc/udev/dev and is restored during system startup.

**ANSWER: B D**

**Explanation:**

Additional rules for udev can be created by adding them to `/etc/udev/rules.d/`. udev processes rules to react to device events and to assign device-node names, permissions, ownership, symbolic links, and other properties. Distribution-provided rule files commonly reside under `/usr/lib/udev/rules.d/` or `/lib/udev/rules.d/`, while `/etc/udev/rules.d/` is the administrator-controlled location for locally created rules and rule overrides. This separation lets local policy persist independently of packaged defaults. The udev rules manual documents the rule search paths and explains that files in `/etc` take precedence when files share a name.

The `/dev/` directory is dynamically populated during system startup and as hardware is detected, rather than being a static collection of nodes for every possible device. In the LPIC-1 context, it is represented as a memory-backed temporary filesystem, allowing device entries to be created and removed dynamically as the system and udev handle device events. This design ensures that device nodes reflect currently available hardware and their configured udev properties. See the udev documentation and the udev rules-file documentation.

## QUESTION NO: 32

Which of the following commands will print important system information such as the kernel version and machine hardware architecture?

**A.** sysinfo

**B.** uname

**C.** lspci

**D.** arch

**E.** info

**ANSWER: B D**

**Explanation:**

`uname` is the standard GNU/Linux utility for displaying system identification information. Depending on its options, it reports the kernel name, kernel release, kernel version, machine hardware name, processor type, hardware platform, and operating system. In particular, `uname -a` produces a broad summary, while `uname -r` displays the kernel release and `uname -m` displays the machine hardware name. This makes `uname` the primary command for obtaining the requested kernel and architecture details.

`arch` prints the machine hardware name, such as `x86_64` or `aarch64`. On GNU systems, its output is equivalent to `uname -m`, so it is a direct and convenient way to identify the machine architecture. These commands are core command-line tools expected for system inspection and basic administration. GNU documentation for `uname` is available in the GNU Coreutils manual; the corresponding documentation for `arch` is at GNU Coreutils: arch invocation.

## QUESTION NO: 33

What do the permissions -rwSr-xr-x mean for a binary file when it is executed as a command?

**A.** The command is SetUID and it will be executed with the effective rights of the owner.

**B.** The command will be executed with the effective rights of the group instead of the owner.

**C.** The execute flag is not set for the owner. Therefore the SetUID flag is ignored.

**D.** The command will be executed with the effective rights of the owner and group.

**ANSWER: A**

**Explanation:**

The command is SetUID and it will be executed with the effective rights of the owner. In the mode string `-rwSr-xr-x`, the uppercase `S` in the owner permission triplet indicates that the set-user-ID bit is set while the owner execute bit itself is not set. The uppercase form describes the displayed permission bits; it does not clear the set-user-ID attribute. This file remains executable by users who qualify through its group or "other" execute permission, because both of those triplets contain `x`. When such a binary is successfully executed, the kernel applies the set-user-ID attribute and sets the process's effective user ID to the UID of the file owner. The process can therefore access resources according to that effective identity, subject to normal system controls such as filesystem mount restrictions and security policy. The GNU Coreutils documentation describes `s`/`S` as the set-user-ID bit, with uppercase `S` denoting that the corresponding execute bit is absent. The Linux `execve(2)` documentation confirms that executing a set-user-ID program changes the effective user ID to the program file's owner. See GNU Coreutils: Structure of File Permissions and execve(2) Linux manual page.

**QUESTION NO: 34**

Which of the following Linux filesystems preallocates a fixed number of inodes at the filesystem's make/creation time and does NOT generate them as needed? (Choose TWO correct answers.)

**A.** ext3

**B.** JFS

**C.** ext2

**D.** XFS

**E.** procfs

**ANSWER: A C**

**Explanation:**

**ext2** and **ext3** preallocate inode tables when the filesystem is created. During `mkfs`, the filesystem layout reserves inode-table blocks for each block group, and the selected bytes-per-inode ratio determines the total number of inodes. Consequently, this inode capacity is fixed for the lifetime of that filesystem: creating additional files consumes entries from the preallocated inode tables, but does not cause new inode tables to be created on demand. This is why an ext2 or ext3 filesystem can run out of inodes even when it still has free data blocks available. Ext3 retains ext2's fundamental on-disk format and adds journaling, so it has the same preallocated inode-table design. The inode density can be chosen at creation time with filesystem-creation settings, making it important to select an appropriate value for workloads expected to contain many small files. The ext2 manual documents inode allocation and creation parameters, while the ext3 documentation describes its compatibility with the ext2 filesystem format: ext2(5) manual page and Linux kernel ext3 documentation.

**QUESTION NO: 35**

Which of the following statements are true about the boot sequence of a PC using a BIOS? (Choose two.)

**A.** Some parts of the boot process can be configured from the BIOS

**B.** Linux does not require the assistance of the BIOS to boot a computer

**C.** The BIOS boot process starts only if secondary storage, such as the hard disk, is functional

**D.** The BIOS initiates the boot process after turning the computer on

**E.** The BIOS is started by loading hardware drivers from secondary storage, such as the hard disk

**ANSWER: A D**

**Explanation:**

Some parts of the boot process can be configured from the BIOS because BIOS setup provides firmware-level settings that control how the machine begins startup. Common configurable settings include the boot-device priority, whether particular storage or removable devices are enabled, and other platform initialization settings. Once power is applied, the processor begins executing the BIOS firmware stored on the system board. The firmware performs initial hardware checks and initialization, commonly including the power-on self-test, then selects a bootable device according to its configured boot order. It loads the initial boot code from that device, transferring control to the boot loader, which can then load the Linux kernel and its associated initial userspace. Therefore, the BIOS initiates the boot process after turning the computer on: its role precedes the operating system and boot loader rather than being started by them. This firmware-driven sequence is the traditional BIOS boot model addressed by LPIC-1 objectives. Red Hat's overview of the boot process describes the firmware stage and the handoff to the boot loader: Red Hat Enterprise Linux System Administrator's Guide. Additional technical background on PC BIOS initialization is available from the Linux kernel x86 boot protocol documentation.

**QUESTION NO: 36**

In compliance with the FHS, in which of the following directories are documentation files found?

**A.** /usr/share/documentation

**B.** /usr/local/share/documentation

**C.** /var/share/doc

**D.** /usr/share/doc

**E.** /etc/share/doc

**ANSWER: D**

**Explanation:**

`/usr/share/doc` is the Filesystem Hierarchy Standard location for documentation associated with installed software. The FHS defines `/usr/share` as the hierarchy for architecture-independent, shareable data. Within that hierarchy, `/usr/share/doc` is designated for miscellaneous documentation and is normally organized into subdirectories for individual packages. For example, a package may install its README files, changelog, copyright or license information, examples, and other supplementary material under a path such as `/usr/share/doc/package-name/`.

This standardized location lets users, package-management tools, and administrators find locally installed package documentation consistently across Linux distributions. Documentation is not necessarily required for every package, but when packaged documentation is installed in accordance with the FHS, `/usr/share/doc` is the appropriate shared location. The FHS also notes that manual pages follow their own standardized paths, such as `/usr/share/man`, while general package documentation belongs beneath `/usr/share/doc`. See the FHS definition of /usr/share and the Linux man-pages documentation for the filesystem hierarchy.

**QUESTION NO: 37**

Which statements about exported Bash variables are correct? (Choose TWO correct answers.)

**A.** `export` makes a variable available to subsequently started child processes.

**B.** An exported variable can still be used by the current shell.

**C.** Every shell variable is exported automatically.

**D.** A child process can permanently change an exported variable in its parent shell.

**E.** Exported variables are stored in `/etc/environment` automatically.

**ANSWER: A B**

**Explanation:**

The `export` builtin marks a shell variable for inclusion in the environment of commands subsequently started by that shell. Therefore, a program launched after `export EDITOR` can read the value of `EDITOR` from its environment.

An exported variable remains a shell variable in the current shell as well. However, changing a variable in a child process cannot modify the parent shell's variable, because child processes receive a copy of the environment when they are created. Bash documents this behavior in its Bourne Shell Builtins reference.