PassQueen
a.k.a dumpsqueen.com

Linux Foundation Certified System Administrator

Linux Foundation LFCS

Version Demo

Total Demo Questions: 36

Total Premium Questions: 363

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Essential Commands	141
Topic 2, Operation of Running Systems	68
Topic 3, User and Group Management	29
Topic 4, Networking	47
Topic 5, Service Configuration	37
Topic 6, Storage Management	33
Topic 7, Mix Questions	8
Total	363

QUESTION NO: 1

Which of the following commands displays the contents of a gzip compressed tar archive?

A. gzip archive.tgz | tar xvf -

B. tar ztf archive.tgz

C. gzip -d archive.tgz | tar tvf -

D. tar cf archive.tgz

ANSWER: B

Explanation:

`tar ztf archive.tgz` displays a listing of the members stored in a tar archive that has been compressed with gzip. In tar's traditional option syntax, `t` selects table-of-contents mode, which lists archive member names rather than extracting them; `z` tells tar to pass the archive through gzip decompression; and `f archive.tgz` specifies the archive file to read. The command therefore reads the compressed tar file, decompresses it as needed, and prints the archive's contents without changing files on disk. This is a common verification step before extraction, particularly when an administrator needs to inspect paths, permissions, ownership, sizes, and timestamps represented in the listing. GNU tar documents `--list` (`-t`) as the operation for listing archive contents and `--gzip` (`-z`) as gzip compression handling. The equivalent long-option form is `tar --gzip --list --file=archive.tgz`. See the GNU tar documentation for listing archive members and its documentation for gzip compression.

QUESTION NO: 2

A user accidentally created the subdirectory \dir in his home directory. Which of the following commands will remove that directory?

A. rmdir '~/\dir'

B. rmdir "~/\dir"

C. rmdir ~/'dir'

D. rmdir ~/\dir

E. rmdir ~/\\dir

ANSWER: E

Explanation:

`rmdir ~/\\dir` is correct because the intended directory name begins with a literal backslash character and is located directly under the current user's home directory. In an unquoted shell word, the leading `~/` undergoes tilde expansion, producing the path to the user's home directory. The two consecutive backslashes that follow are processed by the shell so that the first backslash quotes the second one. Consequently, the command passed to `rmdir` contains one literal backslash in the final pathname: the home-directory path followed by `\dir`.

The `rmdir` utility removes an existing directory when it is empty, which is the normal command to use for a mistakenly created empty subdirectory. Pathname quoting is essential here because backslash has special meaning to the shell: it normally escapes the following character rather than becoming part of the filename. Using an escaped backslash ensures that the filename is addressed exactly as it was created. If the directory contains files or subdirectories, its contents must first be removed or relocated before `rmdir` can remove it. See the GNU Bash documentation on tilde expansion and the GNU Coreutils documentation for rmdir.

QUESTION NO: 3

Which of the following commands will NOT update the modify timestamp on the file /tmp/myfile.txt?

A. file /tmp/myfile.txt

B. echo "Hello" >/tmp/myfile.txt

C. sed -ie "s/1/2/" /tmp/myfile.txt

D. echo -n "Hello" >>/tmp/myfile.txt

E. touch /tmp/myfile.txt

ANSWER: A

Explanation:

`file /tmp/myfile.txt` is correct because it examines the specified pathname to identify the file's type and, where applicable, inspects file contents to determine a more specific description. This is a read-only operation: it does not write data to `/tmp/myfile.txt`, truncate it, replace it, or explicitly set any of its timestamps. Therefore, the file's modification time (mtime)—the timestamp recording when its content was last modified—remains unchanged. Depending on filesystem mount settings and timestamp behavior, reading a file can potentially affect its access time (atime), but atime is distinct from mtime and is not what the question asks about. The `file` utility's documented purpose is to determine file type, and its normal operation is inspection rather than modification. This behavior is appropriate when an administrator needs information about a file without altering its contents or modification timestamp. See the file(1) manual page for the utility's purpose and the inode(7) manual page for Linux file timestamp definitions.

QUESTION NO: 4

Which of the following commands can be used to download the RPM package kernel without installing it?

A. yum download --no-install kernel

B. yumdownloader kernel

C. rpm --download --package kernel

D. rpmdownload kernel

ANSWER: B

Explanation:

`yumdownloader kernel` is designed specifically to retrieve an RPM package from configured YUM repositories without performing an installation. It queries the enabled repositories for the package named `kernel`, resolves the appropriate package version and architecture according to the system's repository configuration, and saves the resulting RPM file in the current working directory by default. Because it is a download utility rather than a package transaction command, it does not alter the installed package set, start services, modify the bootloader, or otherwise change the running system. This is useful when an administrator needs to inspect a package, transfer it to another compatible host, retain it for offline installation, or prepare local installation media. The utility is traditionally supplied by the `yum-utils` package; on distributions using DNF, comparable functionality is provided through the DNF download plugin while compatibility tooling may still expose `yumdownloader`. The command can also be combined with options such as `--destdir` to choose a download directory or `--resolve` when dependent RPMs are also required. See the yumdownloader manual page and the DNF download plugin documentation.

QUESTION NO: 5 - (SIMULATION)

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

ANSWER: See the explanation for the answer

Explanation:

The command is:

`swapoff`

`swapoff` disables swapping on a specified device or swap file when used with its device/file argument, but the question requests only the command name.

QUESTION NO: 6

What is true regarding TCP port 23?

A. Port 23 is the well known port for the telnet service which is a plain text protocol that should no longer be used.

B. Port 23 is the well known port for the SSH service which provides secure logins.

C. Port 23 is the well known port for the rlogin service which is SSL secured by default.

D. Port 23 is the well known port for the system login services which are encrypted when the user runs the starttls command in his login shell.

ANSWER: A

Explanation:

Port 23 is assigned to the Telnet service over TCP. Telnet was designed for remote terminal access and uses a text-based protocol, but it does not provide encryption for the session or credentials. Consequently, usernames, passwords, commands, and session output can be exposed to interception or capture when Telnet is used across an untrusted network. The IANA Service Name and Transport Protocol Port Number Registry lists `telnet` with TCP port 23, establishing the conventional well-known service-port association. Telnet's protocol behavior is specified in RFC 854, which describes its network virtual terminal model rather than a protected transport. Modern Linux administration practice is to avoid Telnet for remote logins and use SSH instead, because SSH provides encrypted communication and server authentication. Telnet may still appear in legacy environments or be used in tightly controlled testing situations, but it should not be enabled for normal remote administrative access. See the IANA service-name and port registry entry for Telnet and RFC 854: Telnet Protocol Specification.

QUESTION NO: 7

Which of the following commands set the sticky bit for the directory /tmp? (Choose TWO correct answers.)

A. chmod +s /tmp

B. chmod +t /tmp

C. chmod 1775 /tmp

D. chmod 4775 /tmp

E. chmod 2775 /tmp

ANSWER: B C

Explanation:

`chmod +t /tmp` sets the sticky bit symbolically, using `t` as the special-mode indicator. This changes the directory's sticky-bit attribute without requiring a numeric mode to be specified. On a shared writable directory such as `/tmp`, the sticky bit restricts removal and renaming of contained files: generally, only the file owner, the directory owner, or root may delete or rename an entry. This is the standard protection expected for temporary directories used by multiple users.

`chmod 1775 /tmp` sets the same sticky bit through numeric permission notation. The leading `1` represents the sticky special permission, while `775` supplies the ordinary owner, group, and other permission bits. Therefore, regardless of the resulting ordinary permissions selected by that command, its leading digit explicitly enables the sticky bit on `/tmp`. GNU coreutils documents both symbolic special-mode syntax and the numeric special-permission digit in its mode structure documentation. The Linux chmod(1) manual page also defines `t` as the sticky-bit permission and describes numeric modes.

QUESTION NO: 8

Which of the following files, when existing, affect the behavior of the Bash shell? (Choose TWO correct answers.)

A. ~/.bashconf

B. ~/.bashrc

C. ~/.bashdefaults

D. ~/.bash_etc

E. ~/.bash_profile

ANSWER: B E

Explanation:

`~/.bashrc` and `~/.bash_profile` are standard per-user Bash startup files, so their presence and contents can change the shell environment and behavior. Bash reads `~/.bashrc` for interactive non-login shells. It is commonly used to define aliases, shell functions, prompt settings, completion behavior, and environment variables that should apply whenever a user opens an interactive terminal. Bash reads `~/.bash_profile` for interactive login shells, such as a console login or an SSH login. This file is typically used for login-session setup, including setting environment variables and starting user-session configuration. A common and useful configuration is for `~/.bash_profile` to source `~/.bashrc`, allowing interactive settings to be shared by both login and non-login interactive shells. These filenames are part of Bash's documented startup-file processing; their effects depend on the kind of shell being invoked and, for login shells, the presence of earlier files such as `~/.bash_profile`. See the GNU Bash manual on startup files and the GNU Bash Reference Manual.

QUESTION NO: 9

Which of the following tasks can be accomplished using the command date? (Choose TWO correct answers.)

A. Synchronize the hardware and system clocks.

B. Output date and time in different formats.

C. Set the system clock.

D. Set the hardware clock.

E. Update the time via NTP.

ANSWER: B C

Explanation:

`date` can output the current date and time in different formats. Its format operand accepts conversion specifications such as `+%F` for an ISO-style date, `+%T` for time, or a custom combination such as `+%Y-%m-%d_%H:%M`. This makes it useful in scripts, logs, reports, and filenames where a specific timestamp representation is needed. GNU Coreutils documents these formatting controls in its date examples.

`date` can also set the system clock when invoked with the `--set=STRING` (or `-s STRING`) option. For example, a privileged administrator can use `date --set="2025-01-15 14:30:00"` to set the kernel-maintained system time. Changing system time requires appropriate administrative privileges because the system clock affects logging, scheduled

jobs, authentication timestamps, and other services. The supported setting syntax and the `--set` option are described in the GNU date invocation documentation.

QUESTION NO: 10

Which of the following commands kills the process with the PID 123 but allows the process to "clean up" before exiting?

A. kill -PIPE 123

B. kill -KILL 123

C. kill -STOP 123

D. kill -TERM 123

ANSWER: D

Explanation:

`kill -TERM 123` sends the SIGTERM signal to process ID 123. SIGTERM is the conventional signal used to request orderly process termination. Unlike an uncatchable forced termination signal, SIGTERM can be caught, handled, or ignored by the target process. A well-designed service or application commonly installs a SIGTERM handler to stop accepting new work, finish or safely cancel current work, close files and network connections, flush buffered data, remove temporary resources, and then exit with an appropriate status. This makes SIGTERM the preferred first step when stopping a process because it gives the application an opportunity to preserve consistency and release resources cleanly. If the process does not terminate after a reasonable grace period, an administrator may then escalate to a forced signal. The `kill` utility accepts symbolic signal names such as `TERM`, and its documentation identifies SIGTERM as the default signal sent when no signal is explicitly specified. See the kill(1) manual page and the signal(7) manual page for signal semantics and process-handling details.

QUESTION NO: 11

Which of the following commands can be used to test DNS name resolution for the host name www.example.com? (Choose TWO correct answers.)

A. getent hosts www.example.com

B. dig www.example.com

C. ip route www.example.com

D. hostnamectl resolve www.example.com

E. ss -n www.example.com

ANSWER: A B

Explanation:

`getent hosts www.example.com` queries the system's configured Name Service Switch sources for host information. When DNS is configured in `/etc/nsswitch.conf`, it can resolve the name using the same name-service configuration that many applications use.

`dig www.example.com` sends a DNS query and displays the response in detail, including answer records when resolution succeeds. It is particularly useful for inspecting DNS server responses, record types, and query status. Both commands can be used to investigate whether a name can be resolved, although their output and resolution paths differ. See the getent(1) manual page and the dig(1) manual page.

QUESTION NO: 12

Which of the following commands lists all defined variables and functions within Bash?

A. env

B. set

C. env -a

D. echo $ENV

ANSWER: B

Explanation:

`set` is correct because, when used without arguments in Bash, it displays the shell's currently defined variables and functions. Its output includes shell variables such as configuration and state values, user-defined variables, and function definitions available in the current shell environment. This makes it useful for inspecting the complete Bash shell state rather than only the exported environment.

Bash distinguishes shell variables from environment variables. Variables become part of the environment only when they are exported, typically with `export`. Since the question asks for all defined variables as well as functions, the shell builtin `set` provides the appropriate comprehensive listing. The Bash Reference Manual documents that invoking `set` with no options or arguments prints the names and values of all shell variables and functions. See the GNU Bash manual: The Set Builtin and the GNU Bash manual: Bash Variables.

QUESTION NO: 13 - (SIMULATION)

After configuring printing on a Linux server, the administrator sends a test file to one of the printers and it fails to print. What command can be used to display the status of the printer's queue? (Specify ONLY the command without any path or parameters.)

ANSWER: See the explanation for the answer

Explanation:

`lpq`

The `lpq` command displays the status of print queues, including queued jobs and printer state.

QUESTION NO: 14

Whichof 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:

`init 1` and `telinit 1` both request SysV init runlevel 1, commonly called single-user or rescue-style maintenance mode. In this runlevel, the system is brought to a minimally operational state intended for administrative recovery and maintenance, rather than operating its usual multiuser services. This reduces the number of active processes and services that could interfere with work such as repairing filesystems, correcting configuration problems, or recovering from boot-related issues.

With SysV init, `init` is the process that manages runlevels, and supplying `1` instructs it to transition to runlevel 1. `telinit` is the administrative interface for communicating a requested runlevel change to the running init process; therefore, `telinit 1` has the same maintenance-oriented result. Administrators should ensure they have appropriate console access before making this transition, because normal network-facing and multiuser services may not be available in the resulting state. The SysV runlevel convention is documented in the init(8) manual page; the relationship between init and telinit is also described in the telinit(8) manual page.

QUESTION NO: 15

How can the existing environment variable FOOBAR be suppressedfor the execution of the script./myscript only?

A. unset -v FOOBAR;./myscript

B. set -a FOOBAR="";./myscript

C. env -u FOOBAR ./myscript

D. env -i FOOBAR./myscript

ANSWER: C

Explanation:

`env -u FOOBAR ./myscript` is the appropriate command because `env` constructs the environment used for the command it launches, and its `-u` option removes the named variable from that environment. Therefore, `./myscript` runs without an environment entry named `FOOBAR`, while the invoking shell retains its existing value afterward. This is especially useful when a variable has been exported and would otherwise be inherited by child processes, but the administrator needs one specific command to execute as though that variable were absent. The space before `./myscript` is required: it separates the variable name supplied to `-u` from the command that `env` must execute. GNU Coreutils documents `env -u` as removing a variable from the environment before running a command: GNU Coreutils: env invocation. The same command form is also described in the Linux `env(1)` manual page: env(1) — Linux manual page.

QUESTION NO: 16

In Bash, inserting 1>&2 after a command redirects

A. standard error to standard input.

B. standard input to standard error.

C. standard output to standard error.

D. standard error to standard output.

E. standard output to standard input.

ANSWER: C

Explanation:

The correct answer is **standard output to standard error.** In Bash, file descriptor `1` denotes standard output (stdout), and file descriptor `2` denotes standard error (stderr). The redirection syntax `1>&2` means “make file descriptor 1 point to the same destination currently used by file descriptor 2.” Therefore, a command such as `command 1>&2` sends its normal output through stderr rather than stdout. This is useful when a script must emit a message as an error, when output needs to be separated from ordinary command results, or when a calling process handles stderr differently from stdout. The `&` is

essential: it indicates that the target, `2`, is a file descriptor rather than a filename. Bash performs redirections in the order they appear, which is especially relevant when combining this form with other redirects such as `2>&1`. The Bash Reference Manual documents duplicating output file descriptors using the `[n]>&word` form, while the GNU Bash manual identifies descriptors 0, 1, and 2 as standard input, standard output, and standard error. See the Bash manual's redirection documentation and the GNU Bash manual.

QUESTION NO: 17

When starting a program with the nice command without any additional parameters, which nice level is set for the resulting process?

A. -10

B. 0

C. 10

D. 20

ANSWER: C

Explanation:

10 is correct in the usual LFCS context, where the command is launched from a shell with the normal niceness of 0. Invoking `nice command` without an option uses the utility's default niceness adjustment of 10. Niceness is a scheduling hint: a higher nice value gives a process lower CPU-scheduling priority relative to processes with lower values. Therefore, starting a command with plain `nice` normally results in a process nice value of 10 rather than the default unmodified value of 0.

The GNU Coreutils documentation describes `nice` as running a command with modified niceness and specifies that the default adjustment is 10 when no adjustment is supplied. The POSIX specification likewise defines a default increment of 10 for the `nice` utility. In practical administration, an explicit adjustment can be supplied with `nice -n N command`; however, no such parameter is present here. Note that niceness is inherited from the parent process, so the default action is technically an increment of 10 from the inherited value, subject to the system's permitted range. Exam questions conventionally assume a normally niced shell, yielding 10. See the GNU Coreutils nice documentation and the POSIX nice specification.

QUESTION NO: 18

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` correctly requests an immediate reboot. The `-r` flag instructs `shutdown` to reboot after the specified shutdown time, and `now` supplies that time explicitly. Before performing the reboot, `shutdown` coordinates an orderly transition: it notifies logged-in users, prevents new logins as appropriate, and asks the init system to stop services and unmount filesystems cleanly. This is the preferred operational approach when a controlled reboot is required.

`telinit 6` is also correct for a SysV-init system because it tells the running `init` process to switch to runlevel 6. In the SysV runlevel convention, runlevel 6 is reserved for rebooting the machine. Init processes the scripts associated with that transition, stopping services according to their configured shutdown ordering before restarting the system. These commands represent the standard SysV-init interfaces for an immediate reboot: one through the shutdown utility and one through an explicit init runlevel change. See the shutdown(8) manual page and the telinit(8) manual page.

QUESTION NO: 19

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 devicesare 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 or character devices in /dev/ using mknod.

D. The /dev/ directory is normally mounted as devtmpfs by the kernel 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/`. This is the standard administrator-managed location for local udev rule files. Rules in this directory can match kernel device events and define actions such as assigning stable symbolic-link names, setting ownership and permissions, applying tags, or running carefully chosen helper actions. Rule files are processed by the udev daemon when devices appear, disappear, or change state, allowing device-node handling to be customized without modifying vendor-supplied rules.

On modern Linux systems, `/dev` is normally mounted as `devtmpfs`, rather than as a conventional persistent disk filesystem. The kernel mounts devtmpfs early during boot and automatically provides device nodes for devices known to the kernel. The udev service then receives device events and applies its rules to manage permissions, ownership, naming, and additional links. Because this filesystem is memory-backed and populated dynamically, its contents are not intended to be restored from a persistent directory at startup. See the udev documentation and the Linux kernel's devtmpfs documentation.

QUESTION NO: 20

Which of the following IPv4 networks are reserved by IANA for private address assignment and private routing? (Choose THREE correct answers.)

A. 127.0.0.0/8

B. 10.0.0.0/8

C. 169.255.0.0/16

D. 172.16.0.0/12

E. 192.168.0.0/16

ANSWER: B D E

Explanation:

The private-use IPv4 ranges are **10.0.0.0/8**, **172.16.0.0/12**, and **192.168.0.0/16**. These are the three address blocks defined by RFC 1918 for use within private networks. They can be freely assigned by an organization for internal hosts, routers, services, virtual machines, and other local systems without obtaining globally unique public IPv4 addresses.

Private-use addresses are intended for routing within an organization or between cooperating private networks. They are not globally unique and must not be advertised as reachable public destinations on the Internet. When systems using these

ranges require Internet access, a gateway commonly performs network address translation, allowing private source addresses to be translated to one or more public addresses. The RFC also makes clear that separate organizations may use the same private ranges independently, which is why address planning and translation are important when networks are interconnected.

The authoritative definition is available in RFC 1918, Address Allocation for Private Internets. IANA's IPv4 Special-Purpose Address Registry also records these blocks as private-use address space.

QUESTION NO: 21

Which of the following fields can be found in the /etc/group file? (Choose THREE correct answers.)

A. The list of users that belong to the group.

B. The home directory of the group.

C. The name of the group.

D. The description of the group.

E. The password of the group.

ANSWER: A C E

Explanation:

The correct fields are **"The list of users that belong to the group."**, **"The name of the group."**, and **"The password of the group."** The traditional `/etc/group` format stores one group per colon-separated line, conventionally represented as `group_name:password:GID:user_list`. The group name is the first field and identifies the group. The password field is the second field; on modern Linux systems it commonly contains `x`, indicating that protected group-password information is maintained in `/etc/gshadow` instead. It nevertheless remains a defined field in the `/etc/group` record format. The final field is a comma-separated list of supplementary members of that group. This member list is used alongside users' primary group assignments, which are recorded through the GID field in `/etc/passwd`. Linux system administration tools such as `groupadd`, `groupmod`, and `gpasswd` manage these group account records and their membership data. See the group(5) manual page for the file layout and the gshadow(5) manual page for protected group-password storage.

QUESTION NO: 22

What is the default action of the split command on an input file?

A. It will break the file into new files of 1,024 byte pieces each.

B. It will break the file into new files of 1,000 line pieces each.

C. It will break the file into new files of 1,024 kilobyte pieces each.

D. It will break the file into new files that are no more than 5% of the size of the original file.

ANSWER: B

Explanation:

`split` normally reads an input file sequentially and creates multiple output files containing 1,000 lines each. This is its default line-count behavior when neither a line-count option such as `-l`/`--lines` nor a byte-size option such as `-b`/`--bytes` is supplied. The generated files use a prefix of `x` by default and alphabetic suffixes, beginning with `xaa`, unless a different prefix or suffix scheme is requested. Thus, "It will break the file into new files of 1,000 line pieces each." correctly describes the default operation. This default is useful for dividing large text files into manageable, line-oriented chunks while keeping each original line intact; a line is not divided between output files under normal text input handling. Administrators can explicitly change the number of lines per output file with `split -l NUMBER FILE`, or instead divide data by byte count

when that is the operational requirement. The GNU Coreutils manual documents that the default number of lines per output file is 1,000. See the GNU Coreutils split invocation manual and the split(1) manual page.

QUESTION NO: 23

What of the following can be done by the command ifconfig? (Choose TWO correct answers.)

A. Set a network interface active or inactive.

B. Specify the kernel module to be used with a network interface.

C. Allow regular users to change the network configuration of a network interface.

D. Change the netmask used on a network interface.

E. Specify which network services are available on a network interface.

ANSWER: A D

Explanation:

`ifconfig` can set an interface administratively up or down. Invoking it with an interface name and the `up` flag activates that interface for use, while the `down` flag deactivates it. This is the traditional net-tools mechanism for controlling an interface's operational administrative state. For example, `ifconfig eth0 up` brings `eth0` up and `ifconfig eth0 down` brings it down.

`ifconfig` can also assign or modify the IPv4 netmask associated with an interface address. The `netmask` argument specifies the subnet mask, such as in `ifconfig eth0 192.0.2.10 netmask 255.255.255.0`. The netmask determines which addresses are considered directly reachable on that local network segment. These capabilities are documented in the ifconfig(8) manual page, which lists `up`, `down`, and `netmask` among the interface configuration parameters. Although modern Linux systems generally prefer the `ip` command from iproute2 for new administration work, `ifconfig` remains able to perform these traditional interface-state and netmask configuration tasks. See also the net-tools project.

QUESTION NO: 24

Which of the following crontab entries will execute myscript at 30 minutes past every hour on Sundays?

A. 0 * * * 30 myscript

B. 30 * * * 6 myscript

C. 30 0 * * 0 myscript

D. 30 0-23 * * 0 myscript

E. 0 0-23 * * 30 myscript

ANSWER: D

Explanation:

`30 0-23 * * 0 myscript` is correct because a standard crontab schedule has five time-and-date fields before the command: minute, hour, day of month, month, and day of week. The minute field, `30`, selects the thirtieth minute of an hour. The hour field, `0-23`, covers every hour in the normal 24-hour clock, from midnight through 23:00. The day-of-month and month fields are wildcards, so they do not impose additional calendar restrictions. Finally, the day-of-week value `0` denotes Sunday in conventional cron implementations; many implementations also accept `7` for Sunday. Therefore, the command runs at 00:30, 01:30, 02:30, and so on through 23:30, but only on Sundays. This schedule is appropriate for a user crontab, provided that `myscript` is executable or is invoked with its required interpreter or absolute path. For the cron field

definitions and day-of-week values, see the crontab(5) manual page and the systemd time and date specification for related scheduling conventions.

QUESTION NO: 25

Which of the following commands creates an ext3 filesystem on /dev/sdb1? (Choose TWO correct answers.)

A. /sbin/mke2fs -j /dev/sdb1

B. /sbin/mkfs -t ext3 /dev/sdb1

C. /sbin/mkfs -c ext3 /dev/sdb1

D. /sbin/mke3fs -j /dev/sdb1

ANSWER: A B

Explanation:

`/sbin/mke2fs -j /dev/sdb1` correctly creates an ext3 filesystem because `mke2fs` creates an ext2-family filesystem and its `-j` option creates an ext3-style journal. The journal is the defining ext3 addition to the ext2 filesystem format, enabling metadata journaling and faster recovery after an unclean shutdown. The `mke2fs` manual explicitly documents `-j` as creating an ext3 journal on the filesystem. See the mke2fs manual page.

`/sbin/mkfs -t ext3 /dev/sdb1` is also correct. The `mkfs` command acts as a front end that selects the filesystem-specific formatter specified by `-t`. Specifying `ext3` causes it to invoke the appropriate ext3 filesystem creation utility for `/dev/sdb1`. This form is useful in scripts and administrative workflows because the requested filesystem type is stated directly. The mkfs manual page documents that `-t` selects the filesystem type to create.

QUESTION NO: 26

Which of the following directives are valid in the `[Timer]` section of a systemd timer unit? (Choose TWO correct answers.)

A. OnCalendar=

B. ExecStart=

C. Unit=

D. WantedBy=

E. Restart=

ANSWER: A C

Explanation:

`OnCalendar=` is valid in a systemd timer unit's `[Timer]` section. It defines calendar-based activation times, such as daily schedules or a specific day and time. For example, `OnCalendar=Mon *-*-* 02:00:00` schedules activation on Mondays at 02:00.

`Unit=` is also valid in the `[Timer]` section. It identifies the unit activated by the timer. If it is omitted, systemd normally activates a service unit with the same base name as the timer. For example, `cleanup.timer` normally activates `cleanup.service`.

`ExecStart=` belongs in a service unit's `[Service]` section, while `WantedBy=` is used in an `[Install]` section for enablement relationships. See the systemd.timer documentation and the systemd.service documentation.

QUESTION NO: 27

Which of the following commands can be used to display the local routing table? (Choose TWO correct answers.)

A. ifconfig

B. dig

C. netstat

D. route

E. traceroute

ANSWER: C D

Explanation:

`netstat` can display the kernel's routing table when used with its routing-related option, commonly `netstat -r` (and often `-n` to prevent address and service-name resolution). This provides a view of routes known locally, including destination networks, gateways, masks, interfaces, and route flags. The `route` command also displays the local kernel IP routing table when invoked without route-modification arguments; `route -n` is commonly used for a numeric, immediately readable result. These commands are traditional Linux networking utilities and remain relevant for interpreting existing system configurations and troubleshooting connectivity. On current Linux systems, the iproute2 command `ip route show` is generally the preferred modern equivalent, but `netstat -r` and `route` directly satisfy the requested task. See the route(8) manual page and the netstat(8) manual page.

QUESTION NO: 28

Which of the following commands connects to the remote host example.com which has OpenSSH listening on TCP port 2222? (Choose TWO correct answers.)

A. ssh --port 2222 example.com

B. ssh -p 2222 example.com

C. ssh -o Port=2222 example.com

D. ssh -o GatewayPort=2222 example.com

E. ssh example.com:2222

ANSWER: B C

Explanation:

`ssh -p 2222 example.com` is correct because the OpenSSH client's `-p` option explicitly sets the destination port used when establishing the SSH connection. The port argument is supplied before the destination host, so this command contacts `example.com` at TCP port 2222 rather than the default SSH port, 22.

`ssh -o Port=2222 example.com` is also correct because `-o` accepts an OpenSSH client configuration option in the same syntax used by `ssh_config`. `Port` is a valid client configuration keyword, and assigning it the value `2222` directs the client to connect to that TCP port for the specified host. This form is especially useful when command-line settings need to mirror configuration-file directives or when several SSH settings are supplied with separate `-o` arguments.

Both commands therefore select the same remote service endpoint: the SSH daemon on `example.com` listening on TCP port 2222. The OpenSSH client documentation describes both the `-p` option and the `Port` configuration keyword at OpenBSD ssh manual and OpenBSD ssh_config Port documentation.

QUESTION NO: 29

Which of the following are tasks handled by a display manager like XDM or KDM? (Choose TWO correct answers.)

A. Start and prepare the desktop environment for the user.

B. Configure additional devices like new monitors or projectors when they are attached.

C. Handle the login of a user.

D. Lock the screen when the user was inactive for a configurable amount of time.

E. Create an X11 configuration file for the current graphic devices and monitors.

ANSWER: A C

Explanation:

A display manager provides the graphical entry point to an X11 desktop system. It presents a login interface, authenticates the user, and initiates that user's graphical session. Therefore, "Handle the login of a user." is a core display-manager responsibility. XDM documentation describes it as managing X displays and providing a graphical login mechanism through its greeter and authentication process.

After successful authentication, a display manager starts the user session according to its configured session scripts and desktop-session selection. This makes "Start and prepare the desktop environment for the user." correct: the display manager launches the selected desktop environment or window-manager session and establishes the session context in which it runs. For example, XDM invokes session startup logic such as `Xsession`; KDE-oriented display managers similarly start the chosen Plasma or other desktop session.

These responsibilities distinguish display managers from the desktop environment itself: the display manager controls graphical login and session startup, enabling the user to reach a ready graphical desktop. See the XDM manual page and the Arch Linux Display Manager documentation for details on graphical authentication and session launching.

QUESTION NO: 30

What is the output of the following command?

```
for token in a b c; do
echo -n ${token};
done
```

A. anbncn

B. abc

C. $token$token$token

D. {a}{b}{c}

E. a b c

ANSWER: B

Explanation:

The output is `abc`. In this shell `for` loop, the variable `token` is assigned each word from the list in sequence: first `a`, then `b`, and finally `c`. On every iteration, parameter expansion replaces `${token}` with that iteration's current value before `echo` runs. Consequently, the loop writes `a`, followed immediately by `b`, then `c`.

The `-n` option tells `echo` not to append its usual trailing newline. Since the loop also does not print spaces or any other separators between iterations, all three values are concatenated on one line as `abc`. The semicolon after `echo -n ${token}` simply terminates that command in the loop body; it does not produce output. This demonstrates standard shell control-flow and parameter expansion behavior used in LFCS-level shell scripting. The Bash Reference Manual documents `for` loops and states that the loop body executes once for each item in the word list: Bash Looping Constructs. For the behavior of `echo -n`, see the Bash builtin documentation: Bash Builtin Commands.

QUESTION NO: 31

In order to display all currently mounted filesystems, which of the following commands could be used? (Choose TWO correct answers.)

A. cat /proc/self/mounts

B. free

C. mount

D. lsmounts

E. cat /proc/filesystems

ANSWER: A C

Explanation:

`cat /proc/self/mounts` displays the mounts visible to the calling process through the kernel's procfs mount information. It includes currently mounted filesystems and provides details such as the source, mount point, filesystem type, and mount options. This procfs entry is especially useful because it represents the process's own mount namespace, which matters on systems using containers or other mount namespaces.

`mount`, when run without arguments, lists currently mounted filesystems. Its output ordinarily shows each mounted source, the directory on which it is mounted, its filesystem type, and applicable mount options. This makes it a standard command-line method for inspecting the active mount table during routine system administration.

Both commands satisfy the requirement to display mounted filesystems and are appropriate knowledge for LFCS-level filesystem administration. For command behavior and options, consult the mount(8) manual page. Kernel documentation on procfs mount information is available in the Linux kernel proc filesystem documentation.

QUESTION NO: 32

Which type of filesystem is created by mkfs when it is executed with the block device name only and without any additional parameters?

A. ext2

B. ext3

C. ext4

D. XFS

E. VFAT

ANSWER: A

Explanation:

`ext2` is the default filesystem type selected by the traditional `mkfs` front-end when it is invoked with only a device pathname and no filesystem-type argument or type-selection option. The command syntax permits a filesystem type to be supplied

explicitly, such as `mkfs -t ext4 /dev/device`, but when that information is omitted, the documented default is `ext2`. In practical administration, administrators normally state the intended type explicitly, often by using a type-specific command such as `mkfs.ext4` or `mkfs.xfs`. Doing so avoids ambiguity, makes automation clearer, and ensures that the filesystem created matches the requirements for features such as journaling, allocation behavior, and compatibility. Nevertheless, for the exact command form in the question—`mkfs` followed solely by a block device—the expected LFCS answer is `ext2`. The util-linux manual page documents both the generic command form and its default filesystem type: mkfs(8) manual page. The broader util-linux documentation is available from the util-linux project.

QUESTION NO: 33

Which of the following commands overwrites the bootloader located on /dev/sda without overwriting the partition table or any data following it?

A. dd if=/dev/zero of=/dev/sda bs=512

B. dd if=/dev/zero of=/dev/sda bs=512 count=1

C. dd if=/dev/zero of=/dev/sda bs=440 count=1

D. dd if=/dev/zero of=/dev/sda bs=440

ANSWER: C

Explanation:

The command `dd if=/dev/zero of=/dev/sda bs=440 count=1` writes exactly one block of 440 bytes of zeroes at the beginning of the disk. On a traditional MBR-partitioned disk, bytes 0 through 439 are the bootstrap code area, where the first-stage bootloader resides. The MBR partition table begins at byte 446 and occupies 64 bytes, followed by the two-byte boot signature. Restricting the write to 440 bytes therefore clears the bootloader code while preserving the disk signature area, partition-table entries, boot signature, and every subsequent sector of disk data. The `count=1` operand is essential because it limits `dd` to a single 440-byte output block. This operation is destructive to the existing MBR boot code, so it should be performed only when intentionally removing or replacing that code and after confirming that `/dev/sda` is the intended target device. GNU dd documents that `bs` sets the input/output block size and `count` limits copied input blocks: GNU Coreutils: dd invocation. The MBR layout, including its 440-byte bootstrap-code region and partition-table offset, is documented by Master boot record.

QUESTION NO: 34

When using rpm --verify to check files created during the installation of RPM packages, which ofthe following information is taken into consideration? (Choose THREE correct answers.)

A. Timestamps

B. MD5 checksums

C. Inodes

D. File sizes

E. GnuPG signatures

ANSWER: A B D

Explanation:

`rpm --verify` compares selected attributes of installed package files against the metadata recorded in the local RPM database when the package was installed. **Timestamps** are included: RPM records and verifies the file modification time, reported by the `T` indicator when it differs. **MD5 checksums** are included in the terminology traditionally used by RPM verification output and LFCS-style exam objectives: the verification process compares the recorded file digest with a newly calculated digest to detect changed file contents. Current RPM implementations may use stronger digest algorithms for

newly built packages, but the underlying verification attribute remains the package-recorded file digest. **File sizes** are also compared to the stored RPM metadata, and a size discrepancy is represented by the `S` verification indicator. These checks allow an administrator to identify installed files that no longer match their packaged state, whether because they were edited, replaced, truncated, or otherwise modified. RPM verification is documented in the RPM manual under the verification operation and its attribute indicators. See the RPM manual page and the Red Hat package-integrity verification documentation.

QUESTION NO: 35

Which of the followingcommand sets the Bash variable named TEST with the content FOO?

A. set TEST="FOO"

B. TEST = "FOO"

C. var TEST="FOO"

D. TEST="FOO"

ANSWER: D

Explanation:

`TEST="FOO"` assigns the string `FOO` to the shell variable named `TEST` in Bash. Shell variable assignments use the form `name=value`, with no whitespace on either side of the equals sign. Quotes are appropriate here because they make the assigned value explicit as a single string and prevent shell expansions from being performed on any special characters that might appear within the value. After executing this assignment, the variable can be read with parameter expansion, such as `echo "$TEST"`, which prints `FOO`. This creates a shell variable in the current Bash process; it is not automatically inherited by programs started from that shell. If inheritance by child processes is required, the variable can subsequently be exported with `export TEST`, or assigned and exported together using `export TEST="FOO"`. Bash documents assignment statements as words in the `name=value` form and specifies that assignment values do not undergo word splitting or pathname expansion. See the GNU Bash Reference Manual: Shell Parameters and the GNU Bash Reference Manual: export builtin.

QUESTION NO: 36

What is the name of the simple graphical login manager that comes with a vanilla X11 installation? (Specify ONLY the command without any path or parameters.)

A. xdm

ANSWER: A

Explanation:

`xdm` is the X Display Manager supplied as part of the traditional X Window System distribution. A display manager runs as a system service, presents a graphical login screen on the local display, authenticates a user, and starts that user's graphical session. Because it is the standard, lightweight display manager associated with a vanilla X11 installation, the command name required to fill the blank is `xdm`, with no directory path or command-line arguments. XDM is configured through files such as `xdm-config` and `Xresources`, and its operation is documented in its manual page. Modern Linux desktop environments may instead use display managers such as GDM, SDDM, or LightDM, but those are desktop- or distribution-oriented alternatives rather than the basic display manager included with the core X.Org X11 environment. The X.Org documentation identifies XDM as the display manager component, and the manual page describes it as a program that manages a collection of X displays, which includes providing graphical login handling. See the X.Org xdm manual page and the X.Org documentation site.