**Chapter 10 System Administration: Core Concepts**

1. How does single-user mode differ from multiuser mode?

A: When a system is in single-user mode, only the system console is enabled. That means you cannot run graphical programs, few of the system daemons are running, and not all of the filesystems are mounted. With the system in multiuser mode you can log in at any terminal or workstation that is set up for login, all of the appropriate filesystems are mounted, and all support services and daemons are enabled and running.

3. What do the letters of the **su** command stand for? (*Hint*: It is not Superuser.)

A: The letters of the **su** stand for substitute user.

What can you do using **su** besides give yourself **root** privileges?

A: **su** can give specific users permission to perform tasks that are normally reserved for a user running with root privileges.

How would you log in as Zach if you did not know his password but knew the root password? How would you establish the same environment that Zach has when he first logs in?

**A:** A user with root privileges can change any user’s password without knowing the old password. To log in as Zach, I can first log in as root, change Zach’s password, and then give the command su Zach, or su – Zach to establish the same environment that Zach has when he logs in.

4. How would you allow a user to execute a specific, privileged command without giving the user the **root** password or permission to use **sudo** to run any command with **root** privileges?

A: You can create a **setuid** (set user ID) file. This program belongs to the only one user who can execute it and has no permissions for other users. Or you can edit the **sudoers** file to allow the user to gain root privileges, grant the user permission to use **sudo** to execute the command.

5. How do you kill process 1648?

A: Give the command **kill –SIGTERM** (kill -15) 1648 to kill process 1648 first. If it does not work, use the command **kill -SIGKILL** (kill -9) for a sure kill.

How do you kill all processes running kmail?

A: Give the command **killall kmail** to send a **SIGTERM** signal to all processes running kmail.

In which instances do you need to work with root privileges?

A: To kill any processes other than ones you own, I must run these commands with root privileges.

6. How can you disble SELinux?

A: there are two ways to disable SELinux.

Method 1: modify the **/etc/sysconfig/selinux** file so it includes the line SELINUX = disabled and reboot the system.

Method 2: use the SELinux tab of the Security Level Configuration window displayed by **system-config-securitylevel** to change the SELinux setting to Disabled.

**Chapter 11 Files, Directories, and Filesystems**

1. What is the function of the **/etc/hosts** file?

A: The **/etc/hosts** file stores the names, IP addresses, and optionally aliases of other systems. Typically, this file holds the hostname and IP address of the local system. It also holds a special entry for localhost.

Which services can you use in place of or to supplement the hosts file?

A: This entry supports the *loopback service*, which allows the local system to talk to itself (e.g., for RPC services). The IP address of the *loopback service* is always 127.0.0.1 (IPv4) or ::1 (IPv6), while 127.0.1.1 names the local system.

2. What does the **/etc/resolv.conf** file do?

A: The **/etc/resolv.conf** file is the resolver configuration file. It provides access to DNS for Internet address resolution.

What do the **nameserver** lines in this file do?

A: The nameserver lines indicate which systems the local system queries to resolve hostnames to IP addresses, and vice versa.

3. What is an inode?

A: An inode is a data structure, stored on disk, that defines a file’s existence and is identified by an inode number. An inode contains critical information about a file, such as the UID of the owner, where it is physically located on the disk, and how many hard links point to it. In addition, SELinux stores extended information about files in inodes.

What happens to the inode when you move a file within a filesystem?

An inode is the control structure for a file. If the two filenames have the same inode number, they share the same control structure and are links to the same file. Hence, when you move a file, including a directory file, within a filesystem, you change the filename portion of the directory entry associated with the inode that describes the file. You do not create a new inode. If you move a file to another filesystem, mv first creates a new inode on the destination filesystem and then deletes the original inode.

4. What does the .. entry in a directory point to?

A: The .. entry is a link to the parent directory.

What does this entry point to in the root (/) directory?

A: In the case of the root directory, there is no parent, so the .. entry is a link to the root directory itself.

5. What is a device file?

A: Device files include both block and character special files and represent device drivers that allow the system to communicate with peripheral devices, such as terminals, printers, and hard disks. And they allow user programs to access hardware devices on the system through the kernel.

Where are device files located?

A: Device files appear in the **/dev** directory and its subdirectories. Each device file represents a device; hence, the system reads from and writes to the file to read from and write to the device it represents.

6. What is a FIFO?

A: A FIFO is a special file, also called a named pipe. You read from and write to the file to read from and write to the pipe.

What does FIFO stand for?

A: The term FIFO stands for first in, first out.

What is another name for a FIFO?

A: a FIFO also called a named pipe.

How does a FIFO work?

A: The first information you put in one end is the first information that comes out the other end.

7. Write a line for the **/etc/fstab** file that mounts the **/dev/sdb1 ext4** filesystem on **/extra** with the following characteristics: The filesystem will not be mounted automatically when the system boots, and anyone can mount and unmounts the filesystem.

A: /dev/sdb1 /extra ext4 user, noauto, rw 0 0

8. Without using rm, how can you delete a file? (Hint: How do you rename a file?)

A: mv file /dev/null

10. Why should **/var** reside on a separate partition from **/usr**?

A: Files in /var change often but files in /usr do not. When a system crashes, it is more likely that a filesystem with recently modified files will become corrupt than a stable filesystem. To lessen the chance of the data in /usr becoming corrupted when a system crashes, keep /usr on a separate partition.

12. How would you mount an ISO image so you could copy files from it without burning it to a CD?

A: mount -t -o loop image.iso /mnt/image