Chapter W26 System Administration – Advanced Topics

Objectives:

\* To further illustrate and follow up on system administration topics covered in Chapter 17.

\* To present download and installation instructions for Webmin.

\* To give another example of system service administration using systemd – vsftpd.

\* To describe the systemd target states.

\* To provide further examples of making hard disk additions to a Linux system with ZFS and mdadm.

\* To completely describe CUPS printing in Linux.

\* To detail other Linux backup facilities and commands, such as rsync and ZFS snapshots.

\* To describe repository management on a Debian-family system.

\* To further describe Linux tasks, processes, and threads.

\* To expand upon systemd journald messaging.

\* To define credentialing and various access control methods, such as DAC, MAC, and RBAC.

\* To describe the sudo program, command, and sudoers file.

\* To illustrate the uses of POSIX.1e and NFSv4 Access Control Lists (ACL’s).

\* To illustrate how to keep a Linux system secure using the Uncomplicated Firewall (ufw).

\* To show various Linux encryption mechanisms, such as encrypting an entire hard disk.

\* To describe process credentialing and Linux process capabilities.

\* To define and give an example of Linux namespaces.

Commands and primitives covered:

ACL addgroup adduser apt-key cat chgrp chmod chown compgen cpio CUPS DAC dd du exec() export fdisk fork() ftp getcap getfacl gpg id journalctl kill lp lpadmin lpc lpinfo lpmove lpoptions lpq lpr lprm lpstat MAC mdadm Mirroring mkfs-ext4 mount NFSv4 nice ping POSIX.1e

process capabilities process credentials ps-aux RAID1 RBAC renice setcap setfacl sudo sudoers file system-config-printer systemctl tar top touch ufw umask umount uname vsftpd wget zfs zpool

W26.1 Introduction

To expand the system administration topics and commands given in Chapter 17 in the printed book, we present more advanced concepts, topics, commands and details in this chapter.

The overriding idea behind system administration of a modern, 21st century Linux system, is the use of systemd to ensure that the Linux kernel works efficiently and effectively to provide computer system concurrency, virtualization, and secure persistence. And this control of the kernel by a “super kernel” must also promote the highest level of system performance and speed, given the use cases the computer might be put to, and the perceived needs of the target user base that the computer serves.

Certainly out of the multitude of possible topics we could have presented, the ones you find basically detailed in Chapter 17, and here, have been selected in somewhat of a subjective way. That selective way was mainly guided by these concerns:

a. The secure maintenance, in terms of concurrency, virtualization, and persistence, of a single Linux computer system that an ordinary novice user can install on her own personal computer.

b. How important the topics are in a perceived ranking of essential Linux system administration tasks.

c. How systemd plays into the maintenance regimen chosen by that ordinary user.

d. The overall pedagogic integration of the selected topics presented on system administration with the other topics and material presented in the printed book, and the “W” chapters here at the book website.

e. How well these topics serve to prepare a student for entry into any chosen Information Technology or Computer Science profession, or how someone already in those professions can use this chapter to better their practice of that profession. In other words, for educational and continuing education audiences.

f. To some degree, making it possible to extrapolate these topics (for audiences in e.) from a single computer system environment to a broader and larger-scaled computing environment, such as is found on small-to-medium sized servers, or to cloud-based, virtual computing. That tactic of extrapolation also guides much of what is found in Chapters W22 through W24 also.

The numbering of the Sections, and Examples in this chapter, proceeds in a mostly-sequential order as they have been selectively extracted from, and cross referenced to, the Sections and Examples in Chapter 17 of the printed book. This selective extraction is done according to the pedagogic needs of the presentation of this advanced topic chapter. They are therefore not numbered here in this chapter in strictly cardinal order. The In-Chapter Exercise, and End-of-Chapter Problem numbering is strictly sequential and in cardinal order.

Anything that you are required to type on the command line is show in **bold** type.

W26.1.2 Webmin Download and Installation

This section details how to install, and do a very basic configuration of Webmin on Linux Mint. Webmin has a GUI control Panel for doing many system administration tasks, and most importantly it is a web based interface which is used to manage your system. Webmin can setup user and group accounts, install web servers, initialize file sharing and may other common system administration activities. Webmin is time-effective for beginners who do not know much about the Linux command line, or who just want a simpler, more intuitive graphical interface..

The example below will lead you through the step-by-step download, installation, and initial minimal but critical configuration of Webmin on Linux Mint.

The instructions given here are a repeat of the instructions given in Appendix A in the printed book.

Example W26.1 Webmin Installation on Linux Mint

Step 1. Add the official repository. You can do this by appending these two lines to your /etc/apt/sources.list file, using the following commands:

$ **sudo vi /etc/apt/sources.list**

Then, add the two lines shown here:

**deb http://download.webmin.com/download/repository sarge contrib**

**deb http://webmin.mirror.somersettechsolutions.co.uk/repository sarge contrib**

Save and exit the file.

Step 2. Fetch and install the GPG key for Webmin, using the following commands:

$ **sudo wget http://www.webmin.com/jcameron-key.asc**

$ **sudo apt-key add jcameron-key.asc**

Step 3. Install Webmin with the following commands:

$ **sudo apt-get update**

$ **sudo apt-get install webmin -y**

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following additional packages will be installed:

apt-show-versions libauthen-pam-perl

The following NEW packages will be installed:

apt-show-versions libauthen-pam-perl webmin

0 upgraded, 3 newly installed, 0 to remove and 45 not upgraded.

Need to get 15.3 MB of archives.

Output truncated…

Step 4. If the ufw is active on your system (which by default on Linux Mint, it is not) allow Webmin through the firewall with the following command:

$ **sudo ufw allow 10000**

Step 4. Webmin uses 10000 as its default port into the system. To access the Webmin panel using your favorite web browser from anywhere on your LAN (or for that matter, from the same machine you installed Webmin on), type this into your URL bar on your web browser.

https://your-ip-address:10000

where your-ip-address is the IP address of the machine you just installed Webmin on.

Step 5. A warning appears in the browser window the first time you try to access Webmin, reading “Your connection is not secure”.

Make the Advanced choice in the dialog box that appears on-screen, and Add Exception button choice in the subsequent box. In the Add Security Dialog Box that appears, click on the Confirm Security Exception button at the bottom of the box.

Step 6. The Webmin Login screen appears in your browser window.

Login to your account on the system with your username and password pair, making sure you put an x in the box for Remember login permanently?.

The Webmin System Information panel appears on screen.

Step 7. The only configuration change we will make for you at this time is to change the init system method used by Webmin from the default Upstart to systemd. Systemd is the modern “super kernel” that controls Linux, which we detail in the chapter on systemd at the Github site.

a. In the webmin panel to the left of the display, make the choice System > Bootup and Shutdown.

b. choose Module Config shown in the upper left of the Bootup and Shutdown display.

c. Under System configuration, make the systemd pull-down menu choice.

d. Make the Save button choice at the bottom-left of Bootup and Shutdown module configuration display.

What this step has achieved for you is allowed Webmin to administrate many operations for system administration using the systemd super-kernel. We show some of these in this chapter, and more completely in the chapter on systemd at the Github site.

8. Play with Webmin, in a non-destructive way. In other words, if you encounter a Webmin function or activity that asks you to change basic configuration of the system, do not proceed at this point. Once you learn more about Linux configuration from the sections in this chapter, you will be better prepared to use Webmin to change them. When you are done, make the Logout choice from the Webmin panel on the left of your web browser display.

As stated above, when you go on to the rest of the sections in Chapter 17 in the printed book, and in this chapter, you can go back into Webmin at any time and experiment with Webmin to find out how relatively efficient and effective Webmin is in handling the tasks we detail in those sections!

Example W26.2 System Service Management Using systemd on Debian-family Linux Mint

Objectives: To install, and start a system service for a secure form of ftp, using systemd

Pre-requisites: Knowledge of basic Linux commands as shown in Chapters 1-13.

Background: One of the most important uses of systemd for system administration is the management of essential system services. vsftpd is a secure ftp server that can be used by client machines on your network or the Internet to log into a host machine, and exchange files. More information about using systemd for service management may be found in Chapter 18 in the printed book.

The instructions for installation of vsftpd given here are a repeat of the instructions in Appendix A in the printed book for a Debian-family Linux Mint system. They also work for Debian and Ubunt systems, since both use the APT package manager. You can also find installation instructions for vsftpd on CentOS 7 systems in Appendix A. We tested the CentOS 7 instructions in Appendix A on CentOS 7.4 and CentOS 7.5 systems.

Requirements: Do the following steps, in the order presented, to meet the requirements of this example:

1. Download and install the vsftpd server with the following command:

$ **sudo apt install vsftpd**

[sudo] password for bob: **QQQ**

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages were automatically installed and are no longer required:

liblxc1 lxc-common lxcfs uidmap

Use 'sudo apt autoremove' to remove them.

The following NEW packages will be installed:

vsftpd

0 upgraded, 1 newly installed, 0 to remove and 321 not upgraded.

Output truncated…

Processing triggers for ureadahead (0.100.0-19) ...

Processing triggers for systemd (229-4ubuntu4) ...

$

2. Use the systemctl command to check the status of the vsftpd service, using the following command:

$ **systemctl status vsftpd.service**

● vsftpd.service - vsftpd FTP server

Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)

Active: active (running) since Sat 2016-07-02 12:10:11 PDT; 2min 15s ago

Main PID: 8780 (vsftpd)

CGroup: /system.slice/vsftpd.service

└─8780 /usr/sbin/vsftpd /etc/vsftpd.conf

Jul 02 12:10:10 bob-VirtualBox systemd[1]: Starting vsftpd FTP server...

Jul 02 12:10:11 bob-VirtualBox systemd[1]: Started vsftpd FTP server.

$

Notice from the above output that the command from step 1. not only downloaded, but also installed and started, the vsftpd service.

3. From another host machine on your network, use the ftp command to connect to the vsftpd server on your machine. Substitute the IP address of the machine you want to ftp into for the IP 192.168.0.30 shown in this command:

# **ftp 192.168.0.30**

Connected to 192.168.0.30.

220 (vsFTPd 3.0.3)

Name (192.168.0.30:bob): bob

331 Please specify the password.

Password: **QQQ**

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

4. Get a directory listing of the files on the machine your are connected to:

ftp> **ls**

*Here is a directory listing on the machine you connected to-*

200 PORT command successful. Consider using PASV.

150

-rw-r--r-- 1 1000 1000 0 Jun 10 08:01 32m

drwxr-xr-x 2 1000 1000 4096 Jul 02 12:13 Desktop

drwxr-xr-x 2 1000 1000 4096 Jun 09 16:51 Documents

drwxr-xr-x 2 1000 1000 4096 Jun 09 16:51 Downloads

drwx------ 23 1000 1000 4096 Jul 02 11:51 Dropbox

drwxr-xr-x 2 1000 1000 4096 Jun 09 16:51 Music

drwxr-xr-x 2 1000 1000 4096 Jun 09 16:51 Pictures

drwxr-xr-x 2 1000 1000 4096 Jun 09 16:51 Public

drwxr-xr-x 2 1000 1000 4096 Jun 09 16:51 Templates

drwxr-xr-x 2 1000 1000 4096 Jun 09 16:51 Videos

-rw-r--r-- 1 1000 1000 134217728 Jun 10 08:14 disk1

226 Directory send OK.

5. Terminate the connection with the following ftp command:

ftp> **exit**

221 Goodbye.

#

6. In order to stop the vsftpd service, use the following command:

$ **sudo systemctl stop vsftpd.service**

$

7. Check the status of the vsftpd service with the following command:

**$** systemctl status vsftpd.service

● vsftpd.service - vsftpd FTP server

Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)

Active: inactive (dead) since Sat 2016-07-02 13:09:59 PDT; 12s ago

Process: 9053 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS)

Main PID: 8780 (code=killed, signal=TERM)

Jul 02 13:09:59 bob systemd[1]: Stopping vsftpd FTP server...

Jul 02 13:09:59 bob systemd[1]: Stopped vsftpd FTP server.

$

8. To restart the vsftpd service, use the following command:

$ **sudo systemctl restart vsftpd.service**

9. To make sure a service starts automatically at boot time, use the following command:

$ **sudo systemctl enable vsftpd.service**

Synchronizing state of vsftpd.service with SysV init with /lib/systemd/systemd-sysv-install...

Executing /lib/systemd/systemd-sysv-install enable vsftpd

$

10. To disable a service from starting at boot, use the following command:

$ **sudo systemctl disable vsftpd.service**

Synchronizing state of vsftpd.service with SysV init with /lib/systemd/systemd-sysv-install...

Executing /lib/systemd/systemd-sysv-install disable vsftpd

insserv: warning: current start runlevel(s) (empty) of script `vsftpd' overrides LSB defaults (2 3 4 5).

insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `vsftpd' overrides LSB defaults (0 1 6).

$

Conclusion: This example allowed you to download, install, and start a system service known as vsftpd. You were then able to connect to and use the computer as an ftp server. Additionally, we showed some basic systemd service management commands applied to the vsftpd service. More information about using systemd for service management may be found in Chapter 18.

W26.2.3.1.1 systemd Bootup

When the system boots, systemd on the kernel image is responsible for initializing the required file systems, services and drivers that are necessary for operation of the system. With systemd, this process is split up into runtime steps whose objectives are target units. (See Section 21.4 for detailed information about target units.) The process is done as much as possible in parallel, and is non-specific, so that the order in which target units are reached is determined at runtime, with some default order.

The following table, Table W26.1, shows some critical systemd target units:

|  |  |
| --- | --- |
| default.target | The target that is booted by default. Not a real target, but rather a symbolic link to another target like graphic.target. |
| emergency.target | Starts an emergency shell on the console. Only use it at the boot prompt as systemd.unit=emergency.target. |
| graphical.target | The default target in a GUI install of Linux Mint. Starts a system with network, multiuser support and a display manager. |
| halt.target | Shuts down the system. |
| multi-user.target | Starts a multiuser system with network. |
| reboot.target | Reboots the system. |
| rescue.target | Starts a single-user system without network. |

Table W26.1 Important systemd Targets

When systemd takes over the boot process, it activates target units that are dependencies of default.target, and all other dependencies. default.target is an alias of graphical.target or multi-user.target, depending on whether the system is configured for a graphical UI or only for a text console.

W26.4.7 Examples of Hard Disk Additions

In the following three examples, we provide an ordinary user, or appointed system administrator, with the techniques necessary to manage various methods of adding hard drives to the system.

In Example W26.8a, we show how to use ZFS to add a new second hard disk to the system, a system that is possibly in conformance with the data storage model we prescribed in Section W26.2.1.

In Example W26.8b, we show how to use the **mdadm** program in Linux to create two USB thumb drives as a mirrored pair. This is a more traditional approach to achieve the safe and efficient archiving of our data.

In Example W26.8c, we show how to install an Ubuntu 18.04 Server Edition Linux system onto two hard disk drives as a mirrored pair to safely archive our data. This is an example of using software RAID1 on the root, or system disk.

Example W26.8a Adding a New Hard Drive Using ZFS

Objectives: To use ZFS to add a new hard disk drive to the system.

Pre-Requisites:

a. Having your storage model conform to the recommendation in Chapter 17, Section 2.1,

b. completion of Chapter 17, Example 17.5, and

c. browsing through Chapter W22 on ZFS at the book website to gain a familiarity with ZFS.

Background: Traditionally, the mounting, addition of a filesystem, and volume management of new hard drives on a Linux system is done by adding entries to the /etc/fstab file, using the mount and mkfs commands, and using LVM. A more contemporary approach is used in this example. You will first install the zfsutils-lunix on your system, and create a new virtual pool to accommodate the new disk. Then, you will take the new hard disk you prepared in Example 17.5 in the printed book, and add it into the new virtual pool as a virtual device. What you will then have accomplished is automatically adding a filesystem and mounting the new hard drive on your system at a default location.

The installation instructions for ZFS given here are a repeat of the instructions given in Appendix A in the printed book.

Requirements: Do the steps below, in the order shown, to complete the requirements for this example.

1. Update your package manager, and download and install the zfsutils-linux on your system, using the following commands:

$ **sudo apt update**

Output truncated...

$ **sudo apt upgrade**

Output truncated...

$ **sudo apt install zfsutils-linux**

Output truncated...

2. Once the zfsutils-linux are installed, list the zpools on your system with the following command:

$ **sudo zpool list**

[sudo] password for bob: **zzz**

no pools available

We can now create a new virtual pool, or zpool, on the disk you added to the system in Example W26.7. Warning: As of the time of writing, zpools can only be created on non-system, or boot, disks in Linux Mint.

3. Create a new zpool on the virtual device (vdev) added in example W26.7, /dev/sdb, with the following command:

$ **sudo zpool create test /dev/sdb**

$

The arguments to the zpool create command are the name of the pool, and the virtual device (or devices) that you want to put in this pool. We named our new pool test. On our system, the logical device name of the hard disk that was originally on the system is sda. The new hard disk added in Example W26.7, as listed in that example, is sdb. We created the new virtual pool on the entire disk, not on any partition on the disk. So now the whole disk is a virtual device in the virtual pool named text.

4. Check the status of the new zpool named test with the following command:

$ **sudo zpool status**

pool: test

state: ONLINE

scan: none requested

config:

NAME STATE READ WRITE CKSUM

test ONLINE 0 0 0

sdb ONLINE 0 0 0

errors: No known data errors

$

5. List the components of the zpool named test with the following command:

$ **sudo zpool list**

NAME SIZE ALLOC FREE EXPANDSZ FRAG CAP DEDUP HEALTH ALTROOT

test 928G 276K 928G - 0% 0% 1.00x ONLINE -

6. Check the zfs filesystem automatically placed on the zpool test with the following command:

$ **sudo zfs list**

NAME USED AVAIL REFER MOUNTPOINT

test 240K 899G 96K /test

Most importantly, the ouput of this command shows that the automatically-created zfs filesystem placed on test is mounted at /test. At this point, you could add zfs filesystems, and user data file in them, along the path /test with the zfs command. We show this in considerable detail in the chapter on ZFS at the Github site.

7. To remove the new virtual pool and the association of /dev/sdb with it, use the following command:

$ **sudo zpool destroy test**

$

8. To erase the zfs metadata from the disk /dev/sdb, use the following command:

$ sudo zpool labelclear -f /dev/sdb

$

Now you can re-partition that disk with Gparrted, and reuse it as a simple ext4 volume if you want to.

Conclusion: An efficient way of implementing our recommended storage model is by adding a new persistent medium, and incorporating that medium with ZFS.

W26.4.7.1 Creating and Managing RAID Arrays in Linux

In order to fully support redundancy in our recommended storage model without using a ZFS component, traditional Linux makes available a Multiple Device Administration (mdadm) program to allow us to securely archive the user data component of that model. Similar to zfs vdevs arrayed in a mirrored zpool, Redundant Array of Inexpensive Disks (RAID) devices are virtual devices created from two or more real block devices, like hard disks. This allows multiple devices, typically whole disk drives, but also partitions on disks, to be combined into a single logical device to hold a single or multiple filesystems. RAID “levels” include various degrees of redundancy to enable the data storage component to survive varying amounts of device failure.

To see a complete description of the mdadm command, see the man page for mdadm on your system. The following is a brief description of mdadm syntax and use:

**mdadm** - manage MD devices or Linux Software RAID

**Syntax: mdadm [mode] <raiddevice> [options] <component-devices>**

where-

**mode** Assemble, Build, Create, Monitor, Grow, Incremental Assembly, Manage, Misc,

and Auto-detect

**<raiddevice>** Name to assign to new virtual device, eg. /dev/md0

**[options]** Mode-selection and other options.

**<component-devices>** The physical devices to be assigned to the array, eg. /dev/sdb

**Output:** New software RAID virtual device, or management of previously-created one.

**Common Options:**

-c, --config= Specify the config file or directory. Default is to use

/etc/mdadm/mdadm.conf and /etc/mdadm/mdadm.conf.d, or if those

are missing then /etc/mdadm.conf and /etc/mdadm.conf.d.

-C --create Create a new array.

-A, --assemble Assemble a pre-existing array.

-G, --grow Change the size or shape of an active array.

**Example:** $ **sudo mdadm --create /dev/md0 --level=mirror --raid-devices=2 /dev/sd[b-c]1**

As superuser, create the array named md0 as a mirror of the 2 first partitions on

physical devices, /dev/sdb and /dev/sdc

Some further examples of **mdadm** syntax and use are as follows:

$ **mdadm --query /dev/name-of-device**

Find out if a given device is a RAID array, or is part of one, and will provide brief

information about the device.

$ **mdadm --assemble --scan**

Assemble and start all arrays listed in the default mdadm config file. This command will typically go in a system startup file.

$ **mdadm --stop --scan**

Shut down all arrays that can be shut down (i.e. are not curb10 currently in use). Typically found in a system shutdown script.

$ **mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sd[ac]1**

Creates /dev/md0 as a RAID1 array consisting of /dev/sda1 and /dev/sdc1.

$ **mdadm -Ac partitions -m 0 /dev/md0**

Scan all partitions and devices listed in /proc/partitions and assemble /dev/md0 out of all such devices with a RAID superblock with a minor number of 0.

$ **mdadm --incremental --rebuild-map --run --scan**

Rebuild the array map from any current arrays, and then start any that can be started.

$ **mdadm /dev/md4 --fail detached --remove detached**

Any devices which are components of /dev/md4 will be marked as faulty and then remove from the array.

$ **mdadm --grow /dev/md4 --level=6 --backup-file=/root/backup-md4**

The array /dev/md4 which is currently a RAID5 array will be converted to RAID6. There should normally already be a spare drive attached to the array as a RAID6 needs one more drive than a matching RAID5.

$ **mdadm --create --help**

Provide help about the Create mode.

$ **mdadm --help**

Provide general help.

$ **mdadm --manage --help**

Provides help topics on the management commands and options of mdadm RAID.

The following set of commands, executed in sequence, allow you to replace a failed disk that is part of a RAID1 array named md0:

$ **mdadm --manage /dev/md0 --add /dev/sdd1**

Adds /dev/sdd1 to the array md0 as a spare disk.

$ **mdadm --manage /dev/md0 --fail /dev/sdb1**

Marking a RAID device, /dev/sdb1, as faulty.

$ **mdadm --manage /dev/md0 --replace /dev/sdb1 --with /dev/sdd1**

Re-add /dev/sdd1 into array /dev/md0, replacing /dev/sdb1.

An Example of Creating a RAID1 Array by Adding New Disks

Using a more traditional approach to adding disks and partitioning them in Linux involves use of command line tools such as fdisk, or gdisk. In addition, to assure redundancy of storage for the user data in our recommended storage model, we can create a level RAID1 mirror using two disks. That way, if one of the disks fails, the data is retained on the other disk in the mirror.

Example W26.8b Adding a RAID1 Mirror Using mdadm

Objectives: To use two USB thumb drives as a mirrored pair to safely archive our data.

Pre-Requisites: Having two identically sized USB thumb drives that are recognized by your system, and available ports on your computer to accommodate them.

Requirements: Do the steps below in the sequence presented to complete the requirements of this example.

The instructions for installing mdadm given here are a repeat of the instructions given in Appendix A in the printed book.

0. Having previously determined that the two thumb drives are recognized on your system, insert them into USB ports. We used two 8 GB Kingston thumb drives, very reliable and always auto-mount on Linux Mint and other NIX-like systems.

On Linux Mint, the two thumb drives were auto-mounted. Use the following commands to unmount them:

$ **sudo umount /dev/sdb1**

$ **sudo umount /dev/sdc1**

1. Install mdadm to your system if it is not there already:

$ **sudo apt-get install mdadm**

Reading package lists... Done

Building dependency tree

Reading state information... Done

Suggested packages:

default-mta | mail-transport-agent

The following NEW packages will be installed:

mdadm

0 upgraded, 1 newly installed, 0 to remove and 54 not upgraded.

Need to get 394 kB of archives.

After this operation, 1,201 kB of additional disk space will be used.

…

W: mdadm: /etc/mdadm/mdadm.conf defines no arrays.

$

2. Examine the USB thumb drives with the following mdadm command:

$ **sudo mdadm -E /dev/sd[b-c]**

/dev/sdb:

MBR Magic : aa55

Partition[0] : 15240575 sectors at 1 (type ee)

/dev/sdc:

MBR Magic : aa55

Partition[0] : 15148607 sectors at 1 (type ee)

3. A critically different operation here: use fdisk one disk at a time to partition the two newly-added USB thumb drives, making sure to first delete any partitions that are on them. For our Kingston USB thumb drives, there was only one default partition on each one, as previously determined in step 0.

$ **sudo fdisk /dev/sdb**

Welcome to fdisk (util-linux 2.27.1).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Command (m for help):**d**

d

Selected partition 1

Partition 1 has been deleted.

Command (m for help):**n**

Partition number (1-128, default 1): **Enter**

First sector (34-15240542, default 2048): **Enter**

Last sector, +sectors or +size{K,M,G,T,P} (2048-15240542, default 15240542): **Enter**

Created a new partition 1 of type 'Linux filesystem' and of size 7.3 GiB.

Command (m for help): **p**

Disk /dev/sdb: 7.3 GiB, 7803174912 bytes, 15240576 sectors

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: gpt

Disk identifier: ABB88BEE-C60C-4B4F-B31B-12089C3DB19E

Device Start End Sectors Size Type

/dev/sdb1 2048 15240542 15238495 7.3G Linux filesystem

Command (m for help): **l**

1 EFI System C12A7328-F81F-11D2-BA4B-00A0C93EC93B

Output truncated…

28 Linux RAID A19D880F-05FC-4D3B-A006-743F0F84911E

Output truncated…

Command (m for help): **t**

Selected partition 1

Hex code (type L to list all codes): **28**

Changed type of partition 'Linux filesystem' to 'Linux RAID'.

Command (m for help): **p**

Disk /dev/sdb: 7.3 GiB, 7803174912 bytes, 15240576 sectors

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: gpt

Disk identifier: ABB88BEE-C60C-4B4F-B31B-12089C3DB19E

Device Start End Sectors Size Type

/dev/sdb1 2048 15240542 15238495 7.3G Linux RAID

Command (m for help): **w**

Repeat Step3 for /dev/sdc

4. Check the metadata superblock on both disks:

$ **sudo mdadm -E /dev/sd[b-c]1**

mdadm: No md superblock detected on /dev/sdb1.

mdadm: No md superblock detected on /dev/sdc1.

$

Notice there are no metadata mdadm superblocks! Basically that means that no RAID arrays have been created yet.

5. Create RAID1 array with the two thumb drives as devices:

$ **sudo mdadm --create /dev/md0 --level=mirror --raid-devices=2 /dev/sd[b-c]1**

mdadm: /dev/sdb1 appears to contain an ext2fs file system

size=7572480K mtime=Wed Dec 14 15:23:01 2016

mdadm: Note: this array has metadata at the start and

may not be suitable as a boot device. If you plan to

store '/boot' on this device please ensure that

your boot-loader understands md/v1.x metadata, or use

--metadata=0.90

mdadm: /dev/sdc1 appears to contain an ext2fs file system

size=7618560K mtime=Wed Dec 14 15:03:37 2016

Continue creating array? **Y**

mdadm: Defaulting to version 1.2 metadata

mdadm: array /dev/md0 started.

$  **cat /proc/mdstat**

Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]

md0 : active raid1 sdc1[1] sdb1[0]

7569152 blocks super 1.2 [2/2] [UU]

[>....................] resync = 3.3% (254912/7569152) finish=33.3min speed=3653K/sec

unused devices: <none>

$ **sudo mdadm -E /dev/sd[b-c]1**

/dev/sdb1:

Magic : a92b4efc

Version : 1.2

Feature Map : 0x0

Array UUID : 0a3bf4bd:be60f78b:5011296b:f9e6ae0c

Name : bob-ProLiant-MicroServer:0 (local to host bob-ProLiant-MicroServer)

Creation Time : Wed Dec 14 15:35:41 2016

Raid Level : raid1

Raid Devices : 2

Output truncated…

/dev/sdc1:

Magic : a92b4efc

Version : 1.2

Feature Map : 0x0

Array UUID : 0a3bf4bd:be60f78b:5011296b:f9e6ae0c

Name : bob-ProLiant-MicroServer:0 (local to host bob-ProLiant-MicroServer)

Creation Time : Wed Dec 14 15:35:41 2016

Raid Level : raid1

Raid Devices : 2

Output truncated…

6. Check RAID device type and RAID array:

$ **sudo mdadm --detail /dev/md0**

/dev/md0:

Version : 1.2

Creation Time : Wed Dec 14 15:35:41 2016

Raid Level : raid1

Array Size : 7569152 (7.22 GiB 7.75 GB)

Used Dev Size : 7569152 (7.22 GiB 7.75 GB)

Raid Devices : 2

Total Devices : 2

Persistence : Superblock is persistent

Update Time : Wed Dec 14 15:39:53 2016

State : clean, resyncing

Active Devices : 2

Working Devices : 2

Failed Devices : 0

Spare Devices : 0

Resync Status : 15% complete

Name : bob-ProLiant-MicroServer:0 (local to host bob-ProLiant-MicroServer)

UUID : 0a3bf4bd:be60f78b:5011296b:f9e6ae0c

Events : 2

Number Major Minor RaidDevice State

0 8 17 0 active sync /dev/sdb1

1 8 33 1 active sync /dev/sdc1

$

7. Wait until the md0 array is totally resynced. On our Linux Mint system, that took approximately 30 minutes for the size of disks we were dealing with. Then, when resyncing is done, create an ext4 File System on md0, and mount it at /mnt/raid1:

$ **sudo mkfs.ext4 /dev/md0**

mke2fs 1.42.13 (17-May-2015)

Creating filesystem with 1892288 4k blocks and 473280 inodes

Filesystem UUID: 68192eb0-dffb-4ab5-a411-973e88d5937d

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done

Writing inode tables: done

Creating journal (32768 blocks): done

Writing superblocks and filesystem accounting information: done

$

8. The following commands mount the array at /mnt/raid1, and allow you to put some data in the filesystem you created there:

$ **sudo mkdir /mnt/raid1**

$ **sudo mount /dev/md0 /mnt/raid1/**

$ **sudo touch /mnt/raid1/linuxthetextbook2.txt**

$ **sudo chmod u+x /mnt/raid1/linuxthetextbook2.txt**

$ **sudo vi /mnt/raid1/linuxthetextbook2.txt**

Add the text-

**Linux: The Textbook 2nd edition RAID setups**

Then save it, and quit vi.

9. (Optional if you want to reuse the thumbdrives for something else after this example is completed)

To auto-mount the RAID1 device when the system boots, use sudo vi to enter the following line at the end of the file /etc/fstab-

/dev/md0 /mnt/raid1 ext4 defaults 0 0

10. (Optional if you want to reuse the thumbdrives for something else after this example is completed)

Save the raid configuration manually to ‘mdadm.conf‘ file.

$ **mdadm --detail --scan --verbose >>** **/etc/mdadm.conf**

11. Test the array by removing one of the thumb drives, and checking again whether the file at

/mnt/raid1/linuxthetextbook2.txt is still there! After pulling one thumb drive out to simulate a disk drive failure, use the following command:

$ **sudo mdadm --detail /dev/md0**

/dev/md0:

Version : 1.2

Creation Time : Wed Dec 14 15:35:41 2016

Raid Level : raid1

Array Size : 7569152 (7.22 GiB 7.75 GB)

Used Dev Size : 7569152 (7.22 GiB 7.75 GB)

Raid Devices : 2

Total Devices : 2

Persistence : Superblock is persistent

Update Time : Wed Dec 14 16:14:45 2016

State : clean, degraded

Active Devices : 1

Working Devices : 1

Failed Devices : 1

Spare Devices : 0

Name : bob-ProLiant-MicroServer:0 (local to host bob-ProLiant-MicroServer)

UUID : 0a3bf4bd:be60f78b:5011296b:f9e6ae0c

Events : 23

Number Major Minor RaidDevice State

0 8 17 0 active sync /dev/sdb1

2 0 0 2 removed

1 8 33 - faulty

$

Notice that the array is “degraded”, and the device /dev/sdc1 has been marked as removed from the array. This has simulated a disk drive failure for the USB thumb drive.

$ **more /mnt/raid1/linuxthetextbook2.txt**

Linux: The Textbook 2nd edition RAID setups

$

But the integrity of the data is maintained on the “Active Device”, /dev/sdb1! This confirms the idea that the data in the RAID1 array named md0 is pread, or mirrored on both the thumb drives, which now act as only a single logical device. That’s because we mirrored the two drives, and made them into one device. Also, files can be accessed via the mount point of this array, /mnt/raid1.

12. To remove the RAID1 array, and zero out the thumb drives for later use, execute the following commands:

$ **sudo umount /dev/md0**

This has unmounted the filesystem on the array.

$ **sudo mdadm --stop /dev/md0**

mdadm: stopped /dev/md0

This has stopped the array.

$ **sudo mdadm --remove /dev/md0**

This has removed the array.

$

Also remove the directory /mnt/raid1, with the command **sudo rm -r /mnt/raid**1,

remove the entry from step 9. in the /etc/fstab file, and delete the /etc/mdadm.conf if you created it in step 10...

Delete the partitions on /dev/sdb and /dev/sdc using fdisk, gdisk, or Gparted, and then create new ones on them formatted to FAT32 if you want to continue using them as USB thumb drives on your system.

Conclusion: This example illustrated the traditional method of adding persistent media to a Linux system, by partitioning it with fdisk, making it redundant using mdadm RAID, creating a filesystem on it, and mounting it at a selected location in the filesystem.

W26.4.7.2 An Example of Installing an Ubuntu 18.04 System on a Mirrored Pair of Hard Disks

To provide integral redundancy of both the operating system itself and user data, it is advisable to install the system in a RAID1 configuration. This is particularly true of systems that must use LXD container virtualization, as detailed in Chapter W23. It allows you to gain all of the advantages provided by software RAID and mdadm. To get the fullest benefit out of this section, we suggest you read through the background material of Section W26.4.7.1 on mdadm and its commands and operations.

Note carefully that the following example is done using the installation of Ubuntu 18.04 Server Edition, on a Linux system capable of having two internally-mounted SATA drives. We present problems at the end of this chapter that ask you to achieve similar results in the other Debian-family and CentOS 7 systems.

Example W26.8c Mirroring of the System Disk and User Data

Objectives: To install a Linux system on two hard disk drives as a mirrored pair to safely archive our data, using software RAID.

Pre-Requisites:

1. Having two identically sized SATA-bus hard drives that are recognized by your system, in our case recognized as sda and sdb.

2. Having the appropriate installation media for Ubuntu 18.04 Server edition as an ISO file on DVD or USB thumb drive.

Background: RAID1, or the ”mirroring” of two or more disks, is primarily a method of using multiple disks to ensure the integrity of your persistent media data by storing it in two or more exact duplicate copies, on each of the disks in the RAID1 array. It can be implemented in software, where the operating system maintains the mirror, or uses special components such as ZFS to achieve the same thing. Or it can be implemented in hardware, where a special RAID controller card on the PCI bus implements the mirroring.

The RAID software included with the current versions of Linux, such as Ubuntu 18.04 Server Edition, is based on the 'mdadm' driver. For desktop systems, and small-to-medium size servers, software RAID is preferable to implementing RAID with hardware RAID controllers.

This Example will guide you through an installation using two RAID1 partitions on two physical hard drives (sda and sdb), each physical drive having one partition used for the root filesystem (/), and another partition for swap space. The method employed has three components: partitioning the two disks appropriately, configuring RAID1 on those disks, and finally formatting the partitions on those disks so that the system can boot from either of them if one or the other should fail.

Requirements: Do the steps below in the sequence presented to complete the requirements of this example, repeating the indicated steps as specified for the swap and root(/) partitions on both drives sda and sdb.

For example, and to clarify the order of steps you must take, in I. Partitioning, you must repeat steps 2. through 5. and 6. through 8. for the swap and root(/) partitions on both drives.

Note that you can move between presented on-screen menu choices using the **Tab** key on your keyboard. Making a menu choice is signified by highlighting it by moving to it, and then pressing **<Enter>**.

I. Partitioning

In order to make the two disks into candidates for a RAID1 configuration, you first have to put partitions on them that will allow you to install the system to those partitions.

Follow the installation steps until you get to the Partition disks screen, then:

1. Select Manual as the partition method. An overview of your available disks is presented in a “Partition disks” screen display with menu choices, such as Guided partitioning, Configure software RAID, Undo changes to partitions, and Finish partitioning and write changes to disk. In our case the disks are displayed as SCSI xxx sda and sdb.
2. Select sda, and choose <Yes> and press **<Enter>** to "Create a new empty partition table on this device".

Repeat this step for sdb.

1. Select the "FREE SPACE" line presented below sda in the “Partition disks” screen display, press **<Enter>**, and then on the “How to use this free space:” screen, select "Create a new partition".
2. Next, enter the size of the partition, in the space “New partition size:”. This partition will be the swap partition. And following a rule of thumb that says swap should be at least twice the size of system RAM, and since our system had 6 Gigabytes of RAM, we entered 12 GB here. Enter the partition size depending on the RAM in your system, choose <Continue>, and press **<Enter>**. On the “type for the new partition:” screen, choose Primary, then press **<Enter>**. On the “Location for the new partition:” screen, choose Beginning, and then press **<Enter>**.
3. The “Partition settings:” screen appears. Select the "Use as:" line at the top. By default this is "Ext4 journaling file system", modify that to "physical volume for RAID". Then select "Done setting up partition", and press **<Enter>**.
4. The overview screen of your disks reappears, as in Step 1. For the root(/) partition, once again select "Free Space" on sda, press **<Enter>** to "Create a new partition".
5. Use the rest of the free space, given the particular drive size you are using, then choose Continue, then select the type for the new partition to be Primary.
6. As with the swap partition, select the "Use as:" line at the top, changing it to "physical volume for RAID". Also select the "Bootable flag:" line to change the value to "on". Then choose "Done setting up partition".
7. Repeat the above steps steps 2. through 5. and 6. through 8. for the other disk and a swap and root partitions on it.

II. Configuring RAID1 using the two disks

With the partitions created properly on both disks, the RAID1 array is ready to be configured using them:

1. Back in the main "Partition disks" screen display, as seen at the start of Requirements Section I., select "Configure Software RAID" at the top.
2. Select "yes" to write the changes to disk.
3. Choose "Create MD device".
4. For this example, select "RAID1".
5. Enter the number of active devices "2", or the amount of hard drives you have, for the array. Then select "Continue".
6. Next, enter the number of spare devices "0" by default, then choose "Continue".
7. Choose which partitions to use. In general, they will be labeled as sda1, sdb1, sdc1, etc. The numbers will usually be the same and the different letters will correspond to different hard drives.

For the swap partition choose sda1 and sdb1. Select "Continue" to go to the next step.

1. Repeat steps II.3. through II.7. for the root(/ )partition, choosing sda2 and sdb2.
2. Select "Finish".

III. Formatting the Drives

The critical operation at this point is to format and set the mount point for the RAID devices.

On the “Partition disks” screen display, there is now a list of hard drives and RAID devices. We need to format and mount the RAID device as a local hard drive, with the following procedures.

1. Highlight "#1" under the "RAID1 device #0" partition.
2. Make the "Use as:" ,emu choice. Then select "swap area", then "Done setting up partition".
3. Highlight "#1" under the "RAID1 device #1" partition.
4. Select "Use as:". Then select "Ext4 journaling file system".
5. Highlight the "Mount point" option, and choose "/ - the root file system". Then select "Done setting up partition".
6. Finally, select "Finish partitioning and write changes to disk".

You want to place the root partition on the RAID array. Therefore, when the installer presents a prompt allow you to boot in a degraded state, reply in the affirmative here. That’s because if one of the array disks fail, you want to be able to boot from the disks in the array that have not failed.

Complete the installation process as you would ordinarily do, making whatever menu choices you need for your particular Linux system installation. After this normal installation, reboot the system, making sure to remove the installation medium when prompted.

* + 1. IV. RAID Status

mdadm commands are used to view the status of an array, add disks to an array, remove disks, and generally manage the disk complement of an array. We show some simple cases of this management, but we encourage you to explore the variety of techniques of array maintenance by using online sources and other references.

To view the status of an array, use the following command-

$ **sudo mdadm -D /dev/md0**

/dev/md0:

Version : 1.2

Creation Time : Thu May 10 22:07:40 2018

Raid Level : raid1

Array Size : 11709440 (11.17 GiB 11.99 GB)

Used Dev Size : 11709440 (11.17 GiB 11.99 GB)

Raid Devices : 2

Total Devices : 2

Persistence : Superblock is persistent

Update Time : Thu May 10 22:11:40 2018

State : clean

Active Devices : 2

Working Devices : 2

Failed Devices : 0

Spare Devices : 0

Name : ubuntu:0 (local to host ubuntu)

UUID : 0b8518fe:66e70e82:f7e98ebe:4160c63c

Events : 17

Number Major Minor RaidDevice State

0 8 1 0 active sync /dev/sda1

1 8 17 1 active sync /dev/sdb1

The -D option of the mdadm command is used to display pertinent information about the /dev/md0 device.

To view the status of a particular disk in the array, use the following command-

$ **sudo mdadm -E /dev/sda1**

/dev/sda1:

Magic : a92b4efc

Version : 1.2

Feature Map : 0x0

Array UUID : 0b8518fe:66e70e82:f7e98ebe:4160c63c

Name : ubuntu:0 (local to host ubuntu)

Creation Time : Thu May 10 22:07:40 2018

Raid Level : raid1

Raid Devices : 2

Avail Dev Size : 23418880 (11.17 GiB 11.99 GB)

Array Size : 11709440 (11.17 GiB 11.99 GB)

Data Offset : 16384 sectors

Super Offset : 8 sectors

Unused Space : before=16232 sectors, after=0 sectors

State : clean

Device UUID : 5f8c6d37:1a2864b3:d84ac0c8:bc810197

Update Time : Thu May 10 22:11:40 2018

Bad Block Log : 512 entries available at offset 136 sectors

Checksum : c6a99aee - correct

Events : 17

Device Role : Active device 0

Array State : AA ('A' == active, '.' == missing, 'R' == replacing)

Conclusion: We illustrated installation of a Linux system on two hard disk drives as a mirrored pair to safely archive our system data and user data, using mdadm software RAID.

W26.5 CUPS Printing

You can configure and manage a printer for use on your Linux system, using the three basic methods we show in this section. These methods are generally applicable to printers that are connected directly to your computer, and this most likely is via a USB connection. We also briefly mention how network-attached printers can be configured.

The three basic methods are:

1. A web-based browser CUPS interface,

2. The built-in system-config-printer utility, and

3. Using the Linux command line.

The configured printer in all three methods is controlled and managed with the Common UNIX Printing System (CUPS).

In method 1, we show a web-based browser front end to CUPS that allows you to manage printers, print jobs, and other configuration settings.

In method 2., we show the built-in graphical front end to CUPS, called system-config-printer, which comes with Linux Mint, and which achieves much of the same functionality as method 1.

In method 3., we show a completely text-based interface for controlling and managing printers from the command line.

W26.5.1 What the Common UNIX Printing System Accomplishes

Using CUPS is a standard way of printing in both Linux and Unix. Since it was developed to provide as many printer definitions as possible, it will more than likely enable you to directly connect your model of printer, or connect to a print server on your LAN.

It is basically composed of two parts- a scheduler, and a filtering system. The scheduler arranges jobs in print queues and sends them to the filtering system that translates the print data into device driver information for the particular printer you want your documents to print on.

W26.5.1.1 Managing CUPS Locally with systemd

Using systemd, via the systemctl command, allows you to start, stop, reload, or restart, the CUPS service. This is a higher level of management for your local printers via a system service. The systemctl command, and its options and arguments, are covered more completely in the chapter on systemd at the Github site.

W26.5.1.2 Starting CUPS Service Using systemd

In Linux Mint, when you attach a new printer via a USB cable to the hardware, it will generally be automatically recognized and attached via CUPS. If the CUPS service has not already been started, it will be automatically started and run in the process of connecting the new printer. But to start the CUPS service without having any printers attached or powered on, do the following:

To start the CUPS service, and check its status, use the following commands:

$ **sudo systemctl start cups**

$sudo systemctl status cups

● cups.service - CUPS Scheduler

Loaded: loaded (/lib/systemd/system/cups.service; enabled; vendor preset: ena

Active: active (running) since Mon 2016-11-07 14:22:24 PST; 1min 40s ago

Docs: man:cupsd(8)

Main PID: 1144 (cupsd)

CGroup: /system.slice/cups.service

├─1144 /usr/sbin/cupsd -l

├─1148 /usr/lib/cups/notifier/dbus dbus://

├─1149 /usr/lib/cups/notifier/dbus dbus://

├─1150 /usr/lib/cups/notifier/dbus dbus://

└─1151 /usr/lib/cups/notifier/dbus dbus://

Nov 07 14:22:24 bob-ProLiant-MicroServer systemd[1]: Stopped CUPS Scheduler.

Nov 07 14:22:24 bob-ProLiant-MicroServer systemd[1]: Started CUPS Scheduler.

Nov 07 14:24:00 bob-ProLiant-MicroServer systemd[1]: Started CUPS Scheduler.

We see from the output that CUPS is running. You can confirm this by looking ahead at Section W26.5.2.

W26.5.1.3 Stopping CUPS service with systemd

The CUPS service can be stopped using the systemctl stop cups.service command, as follows:

$ **sudo systemctl stop cups.service**

When you check the status of the CUPS service after stopping the CUPS service, its status is

inactive (dead), but still enabled.

Enabled means it will persistently start every time the system is rebooted.

W26.5.1.4 Restarting CUPS service with systemd

Restarting a service means that a service is stopped and then started again. If the service

was not currently running, restarting it simply starts the service. Use the following command to restart the CUPS service:

$ **sudo systemctl restart cups.service**

You can also perform a conditional restart of a service using systemctl. A conditional

restart only restarts a service if it is currently running. Any service in an inactive state

is not started.

$ **sudo systemctl condrestart cups.service**

In the above command example, the CUPS service was in an inactive state before the command was executed. When the conditional restart is accomplished, no error messages appear. The cups daemon was not started because conditional restarts only affect active services.

It is always a good practice to check the status of a service, after stopping, starting, or conditionally restarting it.

W26.5.1.5 Configuring CUPS as a Persistent Service Using systemd

You can use the the systemctl command to enable or disable the CUPS services on the Linux server.

Using the enable option on the systemctl command sets a service to always start at

boot (be persistent). The following shows exactly how to accomplish this:

$ **sudo systemctl enable cups.service**

Disabling a service with systemd

You can use the disable option on the systemctl command to keep a service from start-

ing at boot. However, it does not immediately stop the service. You need to use the stop

option discussed in the “Stopping a service with systemd” section. The following example

shows how to disable a currently enabled service.

$ **sudo systemctl disable cups.service**

W26.5.2 Using Web-Based CUPS Administration

CUPS offers its own web-based administrative tool for adding, deleting, and modifying

printer configurations on your computer. The CUPS print service (using the cupsd daemon)

listens on port 631 to provide access to the CUPS web-based administrative interface and

share printers. You can use the CUPS web browser GUI to manage your printing environment, both locally and on a LAN or the Internet, in Linux Mint. This section describes the requirements to use the web browser interface and the administration tasks that you can perform.

W26.5.2.1 Using the Web-Based Interface to CUPS Locally

We found that if a powered-on printer has been automatically detected on your local computer, you can begin to use the web-based browser interface on that local machine. If CUPS has been enabled at system boot, and is running on your computer, you can immediately use CUPS web-based administration from your web browser. A simple and easy way to test whether CUPS is running, is to open a web browser on the local computer and type the following into its

URL locator box:

http://localhost:631

A prompt for a valid login name and password may appear. If so, type the root login name and the root user’s password, and click OK. The web-based CUPS interface Home Tab screen display should appear in your browser window. By default, web-based CUPS administration is available only from the local host.

W26.5.2.1.1 LAN or Web-Based CUPS

To access LAN or web-based CUPS administration from another computer, do the following:

1. On your local machine, from the web-based CUPS interface Home Tab screen, select the Administration Tab. Then put check marks in the boxes next to Allow remote administration, Share printers connected to this system, and Allow printing from the Internet.

2. Select the Change Settings button.

3. You may need to restart the CUPS service with the systemctl command before the change takes effect, as shown in Section W26.5.1.

4. After doing the previous steps, you can view the CUPS interface from a remote computer’s web browser (and you can access CUPS locally as well at localhost:631) using the IP address of the computer you have your printer(s) connected to.

For example, to see the web-based CUPS interface Home Tab screen in a web browser, on a machine with an IP address of 192.168.0.8, type in:

http://192.168.0.8:631

On that machine, you should have a printer detected and powered on.

In-Chapter Exercise W26.1

You have two computers on a LAN, named Proliant and Black\_Dragon. You attach a printer, and it is automatically detected and useable, on Black\_Dragon only. You use the steps shown in Section W26.5.2.1.1 to allow you to manage the printer from your LAN. From a web browser on Proliant, you access the CUPS web-based interface on Black\_Dragon. From Black\_Dragon, can you also use a web browser and access the web-based interface on Proliant to manage the printer on Black\_Dragon?

W26.5.2.1.2 Requirements for Using the CUPS Web Browser Interface

The web-based browser interface can be accessed from a supported browser, such as Firefox. Depending on the task that you are performing, you might be prompted for a user name and password, or for the root user name and password.

The following requirements must be met before using the web-based browser interface:

1. We found that if a printer has been automatically detected on the computer you are trying to access CUPS on, the CUPS daemon goes from the inactive to the active (running) state on that computer. Then you can begin to use the web-based browser interface.

2. The CUPS software packages must be installed on the system that you are accessing via the CUPS web-based browser interface pages. That system can be the local computer, or a remote computer. If you are running Linux Mint, these software packages are installed on your system by default.

3. The following CUPS packages are required:

CUPS, cups-libs, foomatic-db, foomatic-db-engine.

4. The CUPS scheduler, svc:/application/cups/scheduler, must also be running on the system you are accessing.

To verify that the CUPS scheduler is running, open a terminal window and type the following command:

$ **sudo systemctl status cups.service**

[sudo] password for bob: QQQ

● cups.service - CUPS Scheduler

Loaded: loaded (/lib/systemd/system/cups.service; enabled; vendor preset: ena

Active: active (running) since Sat 2016-11-12 16:01:36 PST; 3min 33s ago

Docs: man:cupsd(8)

Main PID: 12086 (cupsd)

CGroup: /system.slice/cups.service

├─12086 /usr/sbin/cupsd -l

├─12091 /usr/lib/cups/notifier/dbus dbus://

└─12092 /usr/lib/cups/notifier/dbus dbus://

Nov 12 16:01:36 bob-ProLiant-MicroServer systemd[1]: Started CUPS Scheduler.

$

W26.5.2.2 Adding a Printer Using the CUPS Web-Browser Interface

To configure a printer that is not automatically detected, you can add a printer from the

Administration Tab seen on the Home Tab screen display. With the Administration Tab screen displayed, you can add a printer as follows:

1. Click the Add Printer button. The Add New Printer screen appears.

2. Check the box that corresponds to the printer you want to add. Then press the Continue button.

3. In the Add Printer dialog box that appears, type a Name, Location, and Description for the printer; also chooses if you want to share this printer, and click Continue.

4. Select the make of the print driver. If you don’t see the manufacturer of your printer listed, choose PostScript for a PostScript printer or HP for a PCL printer. For the manufacturer you choose, you can select a specific model.

5. Choose Add Printer button to continue.

6. On the Set Printer Options page that appears, change any of the default options presented for your printer.

7. Your printer should be available. If the printer is added successfully, click the name of your printer to have the new printer page appear; from the printer page, you can select Maintenance or Administration to print a test page or modify the printer configuration.

W26.5.2.3 Troubleshooting Issues With Accessing the CUPS Web Browser Interface

If you encounter an error while attempting to access the CUPS web browser interface or you cannot access the interface, see Section W26.5.2.1.2 to ensure that all of the requirements have been met. In addition, verify your browser's proxy settings to determine whether a proxy server has been configured. If so, try disabling the proxy server, then re-attempt to access the CUPS web browser interface.

W26.5.2.4 Print Administration Tasks and the Home Tab

Common print administration tasks that you can perform by using the CUPS web browser

interface include the following:

Customizing a print server setup, assigning a print client to a common print server, setting up and managing directly-attached printers and printer classes on servers, setting up and managing remote printers and printer classes on servers, and managing print jobs from print clients

When you first access the CUPS web browser interface at http://localhost:631, you see the menu choices available from the Home Tab screen. From this tab, you can access all of the print administration tasks, which are grouped together by category, as well as the full set of CUPS documentation.

The following tabs are also displayed on the Home Tab screen display:

\* Administration – Enables you to access most print administration tasks, including CUPS

server configuration.

\* Classes – Enables you to search printer classes.CUPS provides collections of printers, which are called printer classes. Print jobs that are sent to a class are forwarded to the first available printer in that class. Classes can be members of other classes. Therefore, you can define very large, distributed printer classes for high-availability printing.

\* Help or Documentation – Enables you to access the CUPS documentation, which includes manuals,

system administration documentation, FAQs, and online help.

\* Jobs – Enables you to view and manage print jobs for configured printers.

\* Printers – Enables you to view information about and modify the settings of a specific printer.

W26.5.2.5 Using the Administration Tab

Most printing tasks can be performed from the Administration Tab.

It is important to realize that some tasks can be performed from multiple tabs.

Basic server settings can also be changed from the Administration Tab. For more information about CUPS server configuration, see the cupsd.conf man page.

The following table, Table W26.3, describes the most important operation categories and individual tasks that can be performed from the Administration Tab.

|  |  |
| --- | --- |
| Operation Category | Task |
| Printers | Add Printer, Find New Printers, Manage Printers |
| Classes | Add Class, Manage Classes |
| Jobs | Manage Jobs |
| Server Edit | Configuration File, View Page Log |

Table W26.3 CUPS Administration Tab Menu Choices

W26.5.2.6 Using the Printers Tab

The Printers Tab, seen on the Home Tab screen display, enables you to view and modify information for configured print queues.

From the Printers tab, you can also perform the following tasks:

Print a test page, stop the printer, reject a print job, move a print job, cancel all print jobs, unpublish the printer, modify a printer configuration, set printer options, delete a printer, set a specific printer as the default, and set allowed users for a printer.

W26.5.2.7 Other Examples of Web-Based Cups Management

With the basic printer configuration done, you can now do further configuration and management of your printers.

Following is a partial listing of important Home Tab screen display menu choices available:

1. List print jobs. Click the Jobs Tab from the Home Tab screen display to see what print jobs are currently active from any of the printers that you have already configured. Click Show Completed Jobs button to see information about jobs that have already printed.

2. Cancel or move a print job. If you sent a print job to the wrong printer, the Move Job selection can be used to move the job to a different printer. From the Administration Tab, click Manage Jobs; then click Show Active Jobs to see what print jobs are currently in the queue for the printer. Select the Cancel Job button next to the print job you want to cancel, or select Move Job to move the print job

to aother printer.

3. Manage Printers. You can click the Manage Printers Tab from the top of the Administration Tab screen display to view the printers you have configured. For each printer that appears, you can select Maintenance or Administrative tasks as follows:

a. Under Maintenance, click Pause Printer (to stop the printer from printing but still

accept print jobs for the queue),

b. Reject Jobs (to not accept any new print jobs),

c. Move All Jobs (to move them to another printer defined on the system),

d. Cancel All Jobs (to delete all print jobs), or Print Test Page (to print a page)

4. Command-line Printing: Select the Command-Line Printing and Options button on the CUPS Home Tab screen display to get help with using Linux command-line methods for printing and doing print management. There is a verbose description of command-line methods shown here.

W26.5.3 CUPS Print Manager GUI

CUPS support in Linux Mint includes a Gnome-based GUI tool for the Cinnamon Desktop, system-config-printer. Generically, across many UNIX and Linux distributions, this tool is known as the Print Manager. It can be launched from the command line, or by making the Linux Mint Menu choice Administration > Printers. CUPS is the default print service on Linux Mint, and is managed by systemd. Detection of directly-attached printers, such as our USB-connected examples in this section, is automatic. CUPS can also automatically discover other CUPS printers on a network, if those printers have sharing enabled. As in other parts of our treatment of CUPS, we do not emphasize the network set of options in this section.

W26.5.3.1 Starting CUPS Print Manager

To start the CUPS Print Manager GUI, use one of the following methods:

From the command line, type the following command:

$ **system-config-printer**

From the Linux Mint Menu, choose Administration > Printers

W26.5.3.2 Setting Up Printers by Using CUPS Print Manager

This section describes procedures and steps that are required to set up a local printer by using CUPS Print Manager. If for some reason you need to either replace/update, reload or install a driver package and any attendant software for your particular printer, skip ahead to Section W26.5.5 for a description of how to do this. We do give indications at some places in the following sub-sections about when and how you would know that the Section W26.5.5 procedures are necessary for your installation and configuration.

W26.5.3.2.1 How to Set Up a New Local Printer

The following example steps give a basic procedure for setting up a new locally-attached printer by using the

CUPS Print Manager GUI. It is possible, given the make and model of the printer you have attached to your Linux Mint computer, that you would only be required to do part of step 1..

Example W26.9 Local Printer Setup

1. Connect the new printer to your local hardware system, then power it on.

If the printer is automatically detected by the system, a notification message appears. If the printer is not automatically detected, there are not many things you can do. This is very similar to adding a new hard disk to the system on the SATA bus: if it is not detected, there are not many trouble-shooting steps you can take.

The notification message displays information about the newly detected printer, and also allows you to print a test page. Since this notification is transitory by default, you can continue on as follows.

You may at this point not have to do any configuration, depending upon whether or not you can actually get a test page, or other document, to print immediately.

a. Start CUPS Print Manager by making the Menu choice Administration > Printers from

or by typing the following command in a terminal window:

$ **system-config-printer**

If you right-click on the newly-attached printer icon display in the Main window of the Print Manager, and make the menu choice Properties, the Printer Properties dialog box appears. On it, there is a button that is labeled Print Test Page. Click on this. The Printer State changes to Processing. If a “good” test page comes out of your printer, game over! You could also attempt to do some further testing by printing document types that you would normally need to print, and verify the results.

If you are adding a new printer that is not automatically configured properly, or you cannot get a test page or other document to print, do the remaining following steps:

b. Choose the Make and Model: Change... button from the Printer Properties window.

2. CUPS selects the USB device that is physically connected to your system.

3. In the Change Driver window, your printer make should be highlighted.

4. Determine whether to accept the default printer driver or provide a PPD file.

To use the default driver, leave the Select Printer From Database option selected.

To provide a PPD file, you can then-

a. Select the Provide PPD File option - The Select a File window is displayed.

b. Locate the specified PPD file on your system, then click Open to associate the PPD file

with the new printer.

5. From the left pane of the next Choose Driver window, select a printer model. From the right

pane, select a printer driver. Then, click Forward.

By default, CUPS selects a “recommended” printer model and the appropriate driver for your printer. You can at this point optionally make another selection from the list of available drivers, if you feel this other driver would work better in configuring the printer.

6. To save your changes, click Apply. If prompted, type the root password.

After you have saved your changes, the newly configured printer is displayed in the CUPS Print

Manager window.

7. (An optional step) To set the printer as the default, right-click the printer name.

a. Choose the Set as Default option.

b. In the Set Default Printer window, choose one of the following options:

Set as the system-wide default printer (default)

Set as my personal default printer

Click OK to save the printer configuration.

8. Try printing a test page, or other document, at the printer. If you are not successful in printing a test page, or another document at the printer, go on to Section W26.5.5 to trouble-shoot your printer. We provide some trouble-shooting techniques in that section. For example, we went through all 8 of the steps above, and could not print a test page! So we went through the trouble-shooting procedures as shown in Section W26.5.5, and were then able to print from the printer.

W26.5.3.2.2 Configuring and Managing Printers by Using CUPS Print Manager

This section describes how to administer printers by using CUPS Print Manager. If you select the printer of interest in the main Print Manager window, and then select Printer > Properties, you are presented with the following choices in the Printer Properties dialog window to modify the properties of a configured printer.

The Printer Properties dialog includes the following six sections for configuring new and

existing printers:

\* Settings

In the Settings section, you can configure the following properties:

Description - Descriptive text about the printer. For example, lpb @ bob-PowerEdge-T110

Location A description of the physical location of the printer - For example, bob-PowerEdge-T110.local

Device URI Information about the protocol that is used to access the printer.

For example, ipps://bob-PowerEdge-T110.local:631/printers/HP\_LaserJet\_P1006

Make and Model Information about the make and model of the printer. For example, Remote Printer.

The default setting for the Make and Model option can be changed by clicking the Change button.

Printer State Information about the current status of the printer. For example, Idle.

Tests and Maintenance Contains the following options:

Print Test Page

Print Self-Test Page

\* Policies

In the Policies section, you can configure the properties that control how a printer behaves.

State

Specifies the following printer states: Enabled, Accepting Requests, Shared

More than one state can be checked off at the same time!

Policies Specifies how the printer behaves during error conditions.

Banner Specifies whether starting or ending banner pages are printed with each print

job.

\* Access Control

The Allow or Deny lists determine which users can print to the printer.

\* Installable Options

For our HP\_LaserJet\_P1006, a Duplexer could be deployed to print on two sides of the same sheet of paper if desired, by checking the Duplexer Installed box.

\* Printer Options

In the Printer Options section, you can configure printer-specific options.

For example, for an HP\_LaserJet\_P1006, the following configurable options are displayed:

Media Size, Double-sided printing, Media source, Output mode, Media type, and Print Quality

The number and types of Printer Options are determined by the PPD file that is associated with the

specified printer.

\* Job Options

Determines the options that are associated with a print job, for example, the number of

copies and page orientation, as well as certain image options. The number and types of

options are determined by the PPD file that is associated with the specified printer.

\* Ink/Toner Levels

Only available if marker levels are reported for this printer.

W26.5.3.2.3 An Example of How to Modify the Properties of an Existing Configured Printer

The following example contains steps that describe how to modify the basic configuration of an existing installed printer. It assumes that you can print from this printer normally.

Example W26.10 Modifying Printer Properties

1. Start the CUPS Print Manager GUI by making the Menu choice Administration > Printers

or by typing the following command in a terminal window:

$ **system-config-printer**

The Printer configuration main dialog window is displayed, and lists all of the configured printers and any newly-detected printers.

2. Right-click the name of the printer for which you want to modify the properties, then choose

Properties.

The Printer Properties dialog box appears, similar to Section W26.5.3.2.2. The Properties dialog box contains six separate sections, each of which contains properties that are grouped by category. By default, the Settings section of the dialog is displayed.

3. In the Settings section, to modify the printer description or location, type any new information that you want in the corresponding text field.

4. A Uniform Resource Identifier (URI) is an addressing technology for identifying resources on the Internet or LAN. The terms URI and URL are used similarly. URIs can be used with application-level protocols, called **URI schemes**. When creating print queues for network-attached printers by using CUPS print commands or the Print Manager, you can specify the device as a **device-uri**.

To modify the device URI:

a. As noted in Section W26.5.3.2.2 our device URI was listed as-

ipps://bob-PowerEdge-T110.local:631/printers/HP\_LaserJet\_P1006

b. Click the Change button next to the setting.

c. From the list of available devices, select a device, then click Apply.

d. When prompted, type the root password. You are then returned to the Settings section.

5. To modify the printer make and model:

a. Click the Change button next to the setting.

b. In the Choose Driver window, select a printer make, then click Forward.

Note – By default, CUPS uses the Select Printer From Database option and selects the

appropriate printer-make for you. Alternatively, you can provide your own PPD file.

c. From the left pane of the next Choose Driver window, select a printer model. From the right

pane, select a printer driver, then click Forward.

d. In the Existing Settings dialog, choose from the following options, then click Apply.

Use the new PPD (Postscript Printer Description) as is.

Try to copy the option settings over from the old PPD.

e. If prompted, type the root password. You are returned to the Settings section of the Printer Properties dialog.

6. Click OK.

W26.5.3.2.4 How to Rename a Printer

It sometimes becomes necessary to rename local printers with more descriptive titles, particularly if you have more than one attached to the system. Of course, renaming network-attached printers is important so that you can keep track of where your documents are being printed.

1. Start the CUPS Print Manager GUI by making the Menu choice Administration > Printers

or by typing the following command in a terminal window:

$ **system-config-printer**

The Printer configuration main dialog window is displayed, and lists all of the configured printers and any newly-detected printers.

2. Right-click the name of the printer that you want to rename.

3. Choose the Rename option.

4. Type a new name for the printer.

5. Type the root password when prompted, if necessary.

6. Click OK to save the changes.

W26.5.3.2.5 How to Duplicate a Printer Configuration

This procedure would be necessary if you wanted to replace an old printer with an exact duplicate, or duplicate an old configuration for a similar new printer.

1. Start the CUPS Print Manager GUI by choosing the Menu Administration > Printers, or by typing the following command in a terminal window:

$ **system-config-printer**

The Printer configuration dialog appears, listing all of the configured printers and any newly-detected printers.

2. Right-click the name of the printer that you want to copy the configuration.

3. Choose the Copy option.

4. In the Copy Printer window, type a name for the printer, then click OK.

5. Type the root password when prompted.

6. Click OK.

W26.5.3.2.6 How to Delete a Printer

This procedure would be necessary when you no longer have a named printer attached to the system, or need to completely replace its configuration definition on the system before upgrading the drivers and other attendant packages related to the printer. For example, in the process of trouble-shooting the HP\_Laserjet\_P1006 in Section W26.5.5, we needed to delete the original printer and its configuration before loading new drivers and installing the printer again.

1. Start the CUPS Print Manager GUI by choosing the Menu Administration > Printers, or by typing the following command in a terminal window:

$ **system-config-printer**

The Printer configuration dialog appears, listing all of the configured printers and any newly-detected printers.

2. Right-click the name of the printer that you want to delete, then choose Delete.

3. Click OK in the Confirm Deletion dialog.

W26.5.3.2.7 How to Disable or Enable a Printer

When you configure a new printer by using CUPS Print Manager, the printer is enabled by

default. This procedure describes how to disable or enable a printer.

1. Start the CUPS Print Manager GUI by choosing the Menu Administration > Printers, or by typing the following command in a terminal window:

$ **system-config-printer**

The Printer configuration dialog appears, listing all of the configured printers and any newly-detected printers.

2. Right-click the name of the printer that you want to disable, or enable, then deselect the option.

3. Type the root password when prompted.

4. Click OK.

W26.5.3.2.8 How to Manage Print Jobs for a Specified Printer

This procedure is probably the most frequent and important one you will do, especially if you are printing high volumes of documents.

1. Start the CUPS Print Manager GUI by choosing the Menu Administration > Printers, or by typing the following command in a terminal window:

**$ system-config-printer**

The Printer configuration dialog appears, listing all of the configured printers and any newly-detected printers.

2. Right-click the name of the printer for which you want to manage print jobs, then choose View

Print Queue.

The Document Print Status (printer-name) window appears, listing all of the print jobs for the

specified printer.

In this window, you can view the following information:

Job, User, Document, Printer size, Time submitted, and Status

3. To view information about completed jobs or printer status, select the appropriate option from

the View menu.

4. To perform a specific action on a print job, select the print job, then select an action from the

available choices on the menubar.

Alternatively, you can right-click the name of a print job, and from the list of available options,

select an action.

5. You can view the following actions:

Cancel, Hold, Release, Reprint

6. (Optional) To refresh the View Print Queue window, choose View → Refresh.

W26.5.4 Configuring and Managing Printers by Using CUPS on the Command-Line

This section provides brief descriptions of some of the CUPS command-line utilities on a Linux system. It describes how to set up and administer your printers with them. We covered the basics of some of these utilities in Chapter 2, and in Chapter 6, Section 6.8. We provide a wider range of examples of command-line control and management of printers in this section.

W26.5.4.1 CUPS Command-Line Utilities

CUPS provides various commands to set up printers and make those printers accessible, both on a local machine where the printer is directly connected to it, and to systems on a LAN or the Internet. In addition, CUPS supports several printer-specific options to the command-line utilities that enable you to control printer configuration. The following table, Table W26.4, lists frequently used CUPS commands.

|  |  |
| --- | --- |
| Command | Task |
| cancel | Cancels a print request |
| lpadmin | Sets up or changes a printer or class configuration |
| lpc | Provides limited control over CUPS print and class queues |
| lpinfo | Shows available devices or drivers known to the CUPS server |
| lpmove | Moves a specified job or all jobs to a new destination |
| lpoptions | Displays or sets printer options and defaults |
| lpq | Shows the current print queue status |
| lp, lpr | Submits a print request |
| lprm | Cancels print jobs that have been queued for printing |
| lpstat | Displays the status information for queues and requests |

Table W26.4 CUPS Command-Line Utilities

Some CUPS command-line names are the same as legacy command-line print commands from UNIX System-V and BSD, but the behavior of commands under CUPS management are somewhat different. You should consult the man pages on your Linux system for all of the commands shown in Table W26.4 for further descriptions and clarification.

For each sub-section below, we first provide the general form of the command-line utility, and then give a specific Example showing actual use of the utility on our Linux Mint system.

W26.5.4.2 How to Set Up a Printer by Using the lpadmin Command

If you have successfully attached a new printer directly to your computer, this section will allow you to view and manage the configuration for that printer.

1. After connecting the printer to the system, turn on the power to the printer.

Consult the printer documentation for information on how to correctly setup the hardware, in terms of USB cables, switch settings on the printer itself, etc..

2. Use the lpadmin command with the -p option to add a printer to CUPS.

Only the most commonly used options of the CUPS lpadmin command are shown here. For information about other options, see the lpadmin man page.

$ **sudo lpadmin -p printer-name -E -v device -m ppd**

where:

-p specifies the name of the printer to add.

-E enables the destination and accepts jobs.

-v sets the device-uri attribute of the print queue.

-m allows designation of the PPD file for the printer, from the “model” directory, or by using

one of the driver interfacesthat your system provides.

3. Verify that the printer is correctly configured.

$ **lpstat -p printer-name -l**

where:

-p the option that specifies you will be providing the name of the printer.

printer-name provides the option argument, which is the actual printer name.

-l shows a long listing of printers, classes, or jobs.

See Example W26.11 for a more practical application of lpadmin command options and their details.

Example W26.11 Specifying a PPD file to Use with the lpadmin Command

To add an HP LaserJet printer, model P1006, by using a network interface URI with the IP address 192.168.0.8, and utilize a particular ppd, you would type the following command:

$ **sudo lpadmin -p HP\_Laserjet\_P1006 -E -v socket://192.168.0.8 -m laserjet.ppd**

W26.5.4.3 Setting a Default Printer on Your System

It is very common on single-computer, and LAN-networked print server configurations, to have a default printer set so that all documents for printing go to that default.

You can specify the default printer used on the system in one of the following ways:

\* By setting the LPDEST or PRINTER environment variable.

The LPDEST environment variable determines the destination of the printer. If the LPDEST

variable is not set, the PRINTER variable is used. The PRINTER variable determines the output

device or destination. If both the LPDEST and PRINTER variables are not set, an unspecified

device is used.

\* By using the lpoptions command.

Use this command to display or set printer options and defaults. For more information, see the lpoptions man page.

The print command searches for the default printer in the following order:

1. The printer name as set by the lp command with the -d option

2. The value of the LPDEST environment variable

3. The value of the PRINTER environment variable

For instructions on setting up printers by using the CUPS web browser interface, see Section W26.5.3.

W26.5.4.4 How to Set a Default Printer at the Command Line

The default printer can be a local printer or a remote printer. Following are various methods of changing the default printer, and some additional Examples that show the application of those methods.

1. Set the system's default printer by using one of the following methods:

\* By specifying the PRINTER variable:

$ **sudo export PRINTER=printer-name**

where printer-name specifies the name of the printer to be assigned as the system's default

printer. If you do not specify printer-name, the system is set up with no default printer.

Note – When using the lp command with the -d option, the destination printer, which might

not be the default printer, is specified. If the -d option is not specified, the print command

searches for information about the printer in the PRINTER environment variable.

\* By specifying the LPDEST variable:

$ **sudo export LPDEST=printer-name**

where:

printer-name specifies the name of the printer to be assigned as the system's default

printer. If you do not specify printer-name, the system is set up with no default printer.

Note – If both the LPDEST and the PRINTER environment variables are set, LPDEST takes

precedence.

\* By using the lpoptions command:

$ **sudo lpoptions -d printer-name**

where:

-d Specifies the destination printer.

printer-name Specifies the name of the printer that is assigned as the system's default printer. If you do not specify printer-name, the system is set up with no default printer.

2. Verify the system's default printer.

$ **lpstat -d**

3. To print to the default printer with the lp command, type the following command:

$ **lp filename**

Example W26.12 Setting a Default Printer by Specifying the PRINTER Variable

The following example shows how to set the printer HP\_Laserjet\_P1006 as the system's default printer by using the PRINTER variable.

$ **export PRINTER=HP\_Laserjet\_P1006**

$ **lpstat -d**

system default destination: HP\_Laserjet\_P1006

Example W26.13 Setting a Default Printer by Specifying the LPDEST Variable

The following example shows how to set the printer HP\_Laserjet\_P1006 as the system's default printer by specifying the LPDEST variable.

$ **sudo export LPDEST=HP\_Laserjet\_P1006**

$ **lpstat -d**

system default destination: HP\_Laserjet\_P1006

Example W26.14 Setting a Default Printer by Using the lpoptions Command

The following example shows how to set the printer HP\_Laserjet\_P1006 as the system's default printer. The printer HP\_Laserjet\_P1006 is used as the system's default printer if the LPDEST or the PRINTER environment variable is not set. Output on our system of the lpoptions command is shown.

$ **lpoptions -d HP\_Laserjet\_P1006**

copies=1 cups-browsed=true device-uri=ipps://bob-PowerEdge-T110.local:631/printers/HP\_LaserJet\_P1006 finishings=3 job-cancel-after=10800 job-hold-until=no-hold job-priority=50 job-sheets=none,none marker-change-time=0 number-up=1 printer-info='lpb @ bob-PowerEdge-T110' printer-is-accepting-jobs=true printer-is-shared=false printer-location=bob-PowerEdge-T110.local printer-make-and-model='Remote Printer' printer-state=3 printer-state-change-time=1478838854 printer-state-reasons=none printer-type=2097158 printer-uri-supported=ipps://bob-PowerEdge-T110.local:631/printers/HP\_LaserJet\_P1006

$ **lpstat -d**

system default destination: HP\_LaserJet\_P1006

The lpoptions command creates a ~/.lpoptions file that includes and entry for the default

printer HP\_Laserjet\_P1006 in the file. By default, all print jobs are now directed to the HP\_Laserjet\_P1006 printer.

W26.5.4.5 How to Print to a Specified Printer

If you have more than one printer defined and directly connected to your system, or you are sharing other network-enabled printers on your LAN, the following steps help you print to one of those specific printers:

1. (An optional step) Verify the status of the printer.

$ **lpstat -p printer-name**

where:

-p is the option allowing you to designate a specific printer.

printer-name is the option argument designating name of the printer you want to print to.

2. Give the destination printer name as an option argument to the lp command.

**$** lp -d destination-printer filename

where:

-d specifies the destination printer.

destination-printer specifies the name of the printer that you are assigning as the

destination printer.

filename specifies the file name to print.

You can also use the lpr command with the -p option to print to a specific printer. For more information, see the lpr man page.

Example W26.15 Printing to a Specified Printer by Using the lp Command

The following example shows how to set the printer HP\_Laserjet\_P1006 as the destination printer when executing the lp command:

$ **lp -d HP\_Laserjet\_P1006 Proposal.doc**

request id is HP\_Laserjet\_P1006-1 (1 file(s))

$ **lpstat -d**

system default destination: HP\_Laserjet\_P1006

The -d option of the lp command takes precedence over the LPDEST and PRINTER environment

variables.

In the above example, the default printer is HP\_Laserjet\_P1006.

W26.5.4.6 How to Verify the Status of Printers

The lpstat command displays information about accessible printers and jobs. Do the following steps to verify the status of printers on your system:

1. Log in to any system on the LAN your computer is hooked up to.

2. (An optional step) Verify the status of all printers, or a specific printer. Only the most commonly used options are shown here. For information about other options, see the lpstat man page.

$ **lpstat [-d] [-p] printer-name [-l] [-t]**

where:

-d shows the system's default printer.

-p printer-name shows that a printer is active or idle, and when the printer was enabled or

disabled.

-l shows the characteristics of printers and jobs.

-t shows status information about CUPS, including the status of all printers,

for example whether printers are active and accepting print requests.

You can specify multiple printer names with this command. Use a space or a comma to separate printer names. If you use spaces, enclose the list of printer names in quotation marks. If you do not specify printer-name, the status of all printers is displayed.

Example W26.16 Displaying the Status of Printers

To display the status of the printer HP\_Laserjet\_P1006, use the following commands:

$ **lpstat -p HP\_Laserjet\_P1006**

printer HP\_Laserjet\_P1006 is idle. enabled since Thu 10 Nov 2016 08:34:14 PM PST

To display the system's default printer, use the following command:

$ **lpstat -d**

system default destination: HP\_Laserjet\_P1006

To display the description of the printers HP-LaserJet-P1200 and HP\_Laserjet\_P1006:

$ **lpstat -p "HP-LaserJet-P1200, HP\_Laserjet\_P1006" -D**

printer HP-LaserJet-P1200 faulted. enabled since Jan 5 11:35 2017. available.

Description: Printer in Orange Bedroom

printer HP\_Laserjet\_P1006 is idle. enabled since Jan 5 11:36 2017. available.

Description: Printer in Basement.

To display the characteristics of the printer HP\_Laserjet\_P1006, use the following command:

$ **lpstat -p HP\_Laserjet\_P1006 -l**

printer HP\_LaserJet\_P1006 is idle. enabled since Jan 5 11:36 2017 PM PST

W26.5.4.7 How to Print a File to the Default Printer

This sub-section is probably the most important and often-used procedure you will do with your computer and printer. Do the following steps:

1. Log in to any system on the network.

2. (Optional) Verify the status of the printer.

$ **lpstat -p printer-name**

3. Issue a print request in one of the following ways:

By using the lp command:

$ **lp filename**

By using the lpr command:

$ **lpr filename**

Note – Only the basic commands are shown in this procedure. For information about the other

Options to these commands, see the lp and lpr man pages on your system, and also refer back to Section 6.8 where there are more examples.

W26.5.4.8 How to Delete a Printer and Remove Printer Access

There may come a time when you want to delete a printer and its configuration completely from the system. For example, when we first installed the HP\_Laserjet\_P1006 on our Linux Mint system, it was erroneously configured with an earlier release of the HPLIP drivers that did not allow us to print with it. So we had to delete that printer and its configuration, and do the trouble-shooting steps shown in Section W26.5.5. We were then able to re-install that printer successfully. Do the following steps to delete a printer, and remove access to it:

1. On the system that is the print client, delete information about the printer.

$ **sudo lpoptions -x printer-name**

where:

printer-name Specifies the name of the printer to delete.

-x Deletes the specified printer.

Note – The -x option only removes the default options for a specific printer and instance. The

original print queue still remains until it is deleted by using the lpadmin command.

In-Chapter Exercise W26.2

Give a general example of using the lpadmin command to delete a print queue.

2. Delete the printer.

$ **sudo lpadmin -x printer-name**

3. Verify that the printer has been deleted, as follows:

a. Confirm that the printer has been deleted on the print client.

$ **sudo lpstat -p printer-name -l**

The command output displays a message indicating the printer does not exist.

b. Confirm that the printer has been deleted on the print server.

$ **sudo lpstat -p printer-name -l**

The command output displays a message indicating that the printer does not exist.

Example W26.17 Deleting a Printer

The following command-line sequence example shows how to delete the printer HP\_Laserjet\_P1006 from the print client named Proliant, and then from the print server named Black\_Dragon.

Proliant$ **sudo lpoptions -x HP\_Laserjet\_P1006**

Proliant$ **sudo lpstat -p HP\_Laserjet\_P1006 -l**

Black\_Dragon$ **sudo lpadmin -x HP\_Laserjet\_P1006**

Black\_Dragon$ **sudo lpstat -p HP\_Laserjet\_P1006 -l**

lpstat: Invalid destination name in list "HP\_Laserjet\_P1006"!

In-Chapter Exercise W26.3

Why is an error generated by the command sudo lpstat -p HP\_Laserjet\_P1006 –l given on the Black\_Dragon?

W26.5.5 Printer Installation and Configuration Troubleshooting

Aside from having hardware-based problems with a new printer you want to add to your system, for example, bad USB cable, laserprinter cartridge improperly installed, etc., one of the most common troubleshooting techniques you can use to get your printer configured is to download and install the necessary drivers for it. Most printer manufacturers make these drivers, and other attendant software, freely available, and in many cases, the Linux kernel already has the necessary driver and software components installed. Most common printers can be automatically attached and configured.

But if you have followed the operations shown in the proceeding sections, and found that your particular printer cannot be automatically recognized or configured, even using the CUPS facilities we have shown, then you can try to reinstall the drivers for it. You need to know what the make and model of the printer is before you begin this troubleshooting operation. To appreciate the procedure involved in doing a typical driver download and installation, follow the specific steps we outline -

1. Search the Internet for the drivers needed for the manufacturer and model of your printer.

For the make and model we have used in our examples in this section, an HP Laserjet P1006, we were able to download (into the Downloads directory) the appropriate drives from the following URL:

http://hplipopensource.com/hplip-web/downloads.html

2. The steps for installing the driver, named hplip-3.16.10.run, for Linux were given at this URL. We then followed the instructions steps, as documented below.

3. We changed the working directory to Downloads, and found the downloaded file there:

$ **cd Downloads**

bob@bob-PowerEdge-T110 ~/Downloads $ **ls**

hplip-3.16.10.run

4. We typed the following command in to begin driver installation:

$ **sh hplip-3.16.10.run**

Creating directory hplip-3.16.10

Verifying archive integrity... All good.

Uncompressing HPLIP 3.16.10 Self Extracting Archive 100%

HP Linux Imaging and Printing System (ver. 3.16.10)

HPLIP Installer ver. 5.1

Copyright (c) 2001-15 HP Development Company, LP

This software comes with ABSOLUTELY NO WARRANTY.

This is free software, and you are welcome to distribute it

under certain conditions. See COPYING file for more details.

Installer log saved in: hplip-install\_Wed-09-Nov-2016\_14:38:44.log

Output truncated…

note: Defaults for each question are maked with a '\*'. Press <enter> to accept the default.

INSTALLATION MODE

-----------------

Please choose the installation mode (a=automatic\*, c=custom, q=quit) : **a**

Initializing. Please wait...

INTRODUCTION

------------

Distro appears to be Linux Mint 18.

Is "Linux Mint 18" your correct distro/OS and version (y=yes\*, n=no, q=quit) ? **y**

ENTER USER PASSWORD

-------------------

Please enter the sudoer (bob)'s password: **QQQ**

INSTALLATION NOTES

------------------

Please read the installation notes. Press <enter> to continue or 'q' to quit: Press Enter on the keyboard

SECURITY PACKAGES

-----------------

Would you like to have this installer install the hplip specific policy/profile (y=yes, n=no\*, q=quit) ? **y**

RUNNING PRE-INSTALL COMMANDS

----------------------------

OK

MISSING DEPENDENCIES

--------------------

Package-Name Component Required/Optional

libcrypto network REQUIRED

Output truncated…

Do you want to install these missing dependencies (y=yes\*, n=no, q=quit) ? **y**

INSTALL MISSING REQUIRED DEPENDENCIES

-------------------------------------

Output truncated…

INSTALL MISSING OPTIONAL DEPENDENCIES

-------------------------------------

note: Installation of dependencies requires an active internet connection.

warning: Missing OPTIONAL dependency for option 'scan': xsane (xsane - Graphical scanner frontend for SANE)

CHECKING FOR NETWORK CONNECTION

-------------------------------

Network connection present.

RUNNING PRE-PACKAGE COMMANDS

----------------------------

Output truncated…

OK

DEPENDENCY AND CONFLICT RESOLUTION

----------------------------------

Output truncated…

HPLIP-3.16.3 exists, this may conflict with the new one being installed.

Do you want to ('i'= Remove and Install\*, 'q'= Quit)?: **i**

Starting uninstallation

HPLIP uninstallation is completed

RUNNING POST-PACKAGE COMMANDS

-----------------------------

OK

RE-CHECKING DEPENDENCIES

------------------------

PRE-BUILD COMMANDS

------------------

OK

BUILD AND INSTALL

-----------------

Output truncated…

HPLIP UPDATE NOTIFICATION

-------------------------

Do you want to check for HPLIP updates?. (y=yes\*, n=no) : **y**

RESTART OR RE-PLUG IS REQUIRED

------------------------------

If you are installing a USB connected printer, and the printer was plugged in when you started this installer, you will need to either restart your PC or unplug and re-plug in your printer (USB cable only). If you choose to restart, run this command after restarting: hp-setup (Note: If you are using a parallel

connection, you will have to restart your PC. If you are using network/wireless, you can ignore and continue).

Restart or re-plug in your printer (r=restart, p=re-plug in\*, i=ignore/continue, q=quit) : **p**

Please unplug and re-plugin your printer now. Press <enter> to continue or 'q' to quit: **<Enter>**

5. At this point, we were able to configure the new printer using a series of dialog boxes very similar to the web-based CUPS GUI methods.

PRINTER SETUP

-------------

Please make sure your printer is connected and powered on at this time.

Do you want to setup printer in GUI mode? (u=GUI mode\*, i=Interactive mode) : **u**

HP Linux Imaging and Printing System (ver. 3.16.10)

Printer/Fax Setup Utility ver. 9.0

Output truncated…

Firmware download successful.

Plug-in installation successful

Done.

Firmware download successful.

Done.

RE-STARTING HP\_SYSTRAY

----------------------

HP Linux Imaging and Printing System (ver. 3.16.10)

System Tray Status Service ver. 2.0

Copyright (c) 2001-15 HP Development Company, LP

This software comes with ABSOLUTELY NO WARRANTY.

This is free software, and you are welcome to distribute it

under certain conditions. See COPYING file for more details.

$

In-Chapter Exercises W26.4 through 13

4. Use the Linux Print Manager utility on your system to add a new printer named “localprinter” to your system (the printer should be connected and powered-on to set up a print queue for the new printer). The printers setup is dependent upon the make and model you have available for your actual system.

For the next four exercises only, power off your printer first. Then use only typed command-line operations to do the next four exercises.

5. Use the lpc command to see the status of all your printers.

6. Use the lpr command to print any particular file of interest to that printer.

7. Check the print queue for that printer to see that the print job is there.

8. Remove the print job from the queue (cancel it).

9. Set up the CUPS web-browser based interface, so other systems on your LAN can print to the printer “localprinter”.

10. Check the status of “localprinter” from a web browser.

11. Actually execute CUPS administration, using the web-browser based interface, from one computer on your LAN to the computer that has “localprinter” connected to it.

12. Use the systemctl command to see the status of the cups.service.

13. Delete the “localprinter” printer from your system, using any of the methods you feel most comfortable with.

W26.6.3 Other Linux Archiving and Backup Facilities

In addition to the traditional tar facility, there are several other facilities and methods a system administrator can use to archive and backup individual user accounts, files, filesystems, and the entire system itself. As stated above, to get a more complete listing of the capabilities and options available for the command line facilities shown in this, and all other sections, consult the man pages on your system for these commands. We briefly describe and give a simple example of some of the more modern and useful of these facilities and methods below. Also, we provide a methodology for doing archiving and backups in Linux using ZFS snapshots. This technique, and the details of working with ZFS, are completely described in the chapter on ZFS at the Github site..

W26.6.3.1 cpio

As universally available as tar on Linux systems, cpio allows the system administrator to backup the entire system and transfer files between file systems. It may be used in conjuction with the **find** command, but not necessarily if you are backing up an entire file system. An abbreviated listing of the man page for cpio is as following-

cpio

Purpose: Copies files to an archive, extracts files from an archive, or passes files to another directory tree.

Syntax:

**cpio -o [aBcv] >** **directory** for creation of an archive, and

**cpio -i [Btv] [pattern]** for restoring an archive.

Output: Created or Restored archive file.

Common Options:

-o creates the archive

-v prints the names of the files that are archived

-i extracts the archive

Command Arguments:

directory A directory where the archive is found.

pattern Source for the restored archive.

When creating an archive, cpio takes the list of files to be processed from the standard

input, and then sends the archive to the standard output. A simple example of this would be as follows:

$ **ls | cpio -ov > backup.cpio**

In a more complicated example, the following commands backup selected files in the /home/bob directory to a USB thumbdrive, generically named **device**-

$ **cd /home/bob**

$ **touch level.1.cpio.timestamp**

$ **find . –newer level.0.cpio.timestamp –print \ | cpio –oacvB > device**

Output truncated…

$

A simple example of extracting a cpio archive is as follows:

$ **cpio -iv < backup.cpio**

W26.6.3.3 dd

The **dd** facility is used to copy a single file, part of a file, a partition, or part of a partition, and can treat the data stream using, for example, compression or format conversion.

An abbreviated version of the dd man page is as follows:

dd

Purpose: To copy a file (from standard input to standard output, by default) with possibly a changeable I/O block size, while optionally doing file conversions on it.

Syntax:

**dd [options] if=device of=device bs=blocksize**

Output:

Modified file.

Common Options:

--help Provides help on the dd command.

--version Supplies the version number of the utility.

Operands and Command Arguments:

ibs=bytes  Sets the input block size in bytes. This makes dd read bytes per block.

The default is 512 bytes.

obs=bytes  Sets the output block size to bytes. This makes dd write bytes per

block. The default is 512 bytes.

bs=bytes  Set both input and output block sizes to bytes. This makes dd read

and write bytes per block, overriding any ‘ibs’ and ‘obs’ settings.

device: file or pathname to object being processed.

blocksize Blocksize of copied file.

A multi-command example of using **dd**, used in conjunction with **ssh** and **tar** , is as follows:

$ **ssh bob@192.168.0.13:/home/bob “dd if=backup.tar ibs=512” | tar --extract --verbose --read-full-records --file –**

The above command extracts the remote file backup.tar file at 192.168.0.13:/home/bob in input 512 byte blocks and streams it through **dd** to the system you typed this command on.

W26.6.3.4 rsync

As shown and detailed in Chapter 17, Section 6.3.1, the **rsync** command is a very space-efficient way to backup selected files and directories, particularly from one machine to another using **ssh** across a network. Its operation can also be automated via the use of systemd scheduling “timers”, a topic illustrated in Chapters 18 and W27. Here we provide an extensive example of the use of **rsync**, coupled with Python, to underscore our point about the utility of both **rsync** and Python usage in Linux.

Example W26.19 Extended Python Script Example Using **rsync** to do a “Rotating” Backup

Objective: Use Python3 in conjunction with the **rsync** command to do a daily, rotating backup of selected directories and files locally and across a network, with a depth of 5 sequentially retained backups.

Pre-Requisites: Reading through Chapter 17, Section 6.3.1, and Chapter W19, Section 4.2.2 for understanding, and doing the Examples W19.31 through W19.34 found there. This preparation will allow you to gain some familiarity with what the example presented here is based upon.

Additionally, you should prepare for executing the Python script file by doing the following-

a. Create a file in /home/your\_username/.rsync/exclude (where your\_username is your login name on your Linux system) which contains filename matching patterns for **rsync** to ignore. You should exclude patterns such as \*.tmp or \*.o

b. You have a remote machine on your LAN or intranet, and know its IP address, that you can ssh into. You must have the access permissions and credentials to access directories and files on that remote machine. On our Linux system, this IP address was 192.168.0.25.

c. You should make sure that you setup ssh on the remote machine so that you can do a login to it without having to type in a password. Not having done this would prevent automating the Python script with legacy cron, or a systemd timer. We provide a Project at the end of this chapter that asks you to automate this script file using a systemd timer.

We provide the Python code for the example presented here at the GitHub site for this book, under the code materials for Chapter W26.

Background: This example is similar to Example W19.34 in Chapter W19 on Python at this book’s website, with the notable exceptions that it is much more extensive, and copies more hard linked files and directories across a network using ssh. As well as using the **rsync** command to achieve the objectives, it also uses the **cpio** command and the Python methods copytree and rmtree, from the shutil utility.

Basically there are five operations this Python script file performs-

1. Checks to see if several numbered, or versioned, backup directories exist. If they don't exist, it creates them.

2. Removes the last version, or oldest directory.

3. Hard copies directory "1" into directory "2". A hard copy makes a copy in which the two files share the same disk space. i.e. your files take up no extra room.

4**. rsync**’s the files you want to backup to directory "1". **rsync** only overwrites changed files. This example does this **rsync** operation first between source and local machine “backup “directories, and then between those backup directories, and their equivalent directories and files on the remote machine.

5. Also, specifically backs up designated source code, a very critical operation if you’re a developer of applications.

Notice that the provided script file does not use OOP, but uses an imperative, procedural programming methodology. We have addressed this issue in Chapter W19.

Requirements:

1. Download the Python source code for this example from the Chapter W26 code materials provided at the book GitHub site.

2. Do the Pre-Requisites carefully, until you have mastered the details of them, and have accomplished what they ask for. In particular, try to comprehend the sequence of the script file as it accomplishes its procedures from beginning to end. Produce a diagram or flowchart of how the program works.

3. \*\*Critical\*\* Modify the downloaded Python source code using your favorite text editor, so that the lines of code under the function backupserver(), that assigns sources, localsources, localcode, target, host (the remote machine), and user are relevant and pertinent to directories, host, and your username on your Linux system.

4. Once the Step 3. change are made, execute your modified Python code.

5. Verify that the script file is doing what it is intended to do; in other words, test it on your Linux system. That testing will include debugging, and a verification regimen that is your responsibility to design and certify.

Conclusion: Given your Linux system administration backup requirements, this example has provided a technique you can deploy, and that uses a powerful strategy, for fairly complex archiving of your system’s persistent data.

W26.6.3.7 Using ZFS Snapshots to Backup File Systems

If you have installed the kernel-loadable model for ZFS on your Linux system, as we detail in the Chapter W22 on ZFS at the book website, you can then use a powerful facility in ZFS, known as “snapshots”, to archive and backup your data. In particular, Section W22.4 details the basic mechanics of creating and managing ZFS snapshots. We also provide an example of doing ZFS snapshots, Example W22.7, which couples ZFS snapshots with a Bash script file for doing incremental backups of a local file system to a remote system.

Of course, this strategy is controlled by exactly what you want to archive and backup. Using ZFS along with our recommended storage model, where the operating system is installed on an ssd, and all of your user data files are stored on another persistent medium, such as additional ssd’s or hard disks, allows you a very time and resource efficient way of doing archiving and backup.

W26.7.3 Repository Management

Since Linux Mint is derived from Ubuntu and Debian Linux, a large majority of its default software, in the form of pre-configured packages, comes from those sources. The repositories are grouped into categories according to how much support is given to packages in a particular category. For example, the main Linux Mint repository contains packages that they directly have a hand in writing source code and other attendant modules for. Unsupported, and to some degree unreliable software packages, are available in the other major categories, Universe and Multiverse. There is always the possibility, with an “open” software operating system such as Linux, to install from source code itself. But this route has largely been co-opted by various levels and forms of pre-packaged applications.

As was seen in Section W26.1.1, is is possible to install from alternative software repositories in order to be able to install software not found at the default Main, Universe, or Multiverse repositories. We repeat the procedure here for explanatory purposes. Be aware that at the time of the writing of this book, Webmin could have been installed using the Software Manager, but we found that way of doing the installation to be unreliable. To install Webmin reliably it is necessary for you go through the following steps, which are a repeat of the installation instructions in Appendix A, Section W26.1.2, of the printed book:

Step 1. Add the official repository. You can do this by appending these two lines to your /etc/apt/sources.list file, using the following commands:

$ **sudo vi /etc/apt/sources.list**

Then, add the two lines shown here into that file:

**deb http://download.webmin.com/download/repository sarge contrib**

**deb http://webmin.mirror.somersettechsolutions.co.uk/repository sarge contrib**

Save and exit the file.

Step 2. Fetch and install the GPG key for Webmin, using the following commands:

$ **sudo wget http://www.webmin.com/jcameron-key.asc**

$ **sudo apt-key add jcameron-key.asc**

Step 3. Install Webmin with the following commands:

$ **sudo apt-get update**

$ **sudo apt-get install webmin -y**

As can be seen in graphical form with the Synaptic Package Manager, the four main repositories that you can pull software from are:

Main - Supported free and open-source software.

Universe - Community-maintained free and open-source software.

Restricted - Proprietary drivers for devices.

Multiverse - Software restricted by copyright or legal issues.

GUI-based repository management, in other words, where you get your software packages from, is normally accomplished via the Linux Mint Menu choice Administration > Software Sources. We have also shown GUI-based management above with the Software Manager and the Synaptic Package Manager. But underneath these GUI’s, APT is the primary tool for installing packages. In the following section, we show some of the basic characteristics of package repository listings, and the files that they give you access to.

W26.7.3.1 Basic Repository Characteristics

APT stores a list of repositories, sometimes called “software channels”, in the file /etc/apt/sources.list.

This is the file you edited above to add the Webmin repositories as a spurce for where you obtain packages from. It also stores lists of repositories in any file with the suffix .list under the directory /etc/apt/sources.list.d/

You can add new repositories to the /etc/apt/sources.list as we showed above, but APT

will also, by default look in the /etc/apt/sources.list.d/ directory for text files ending with the .list

extension. In order to take effect, the lines in these text files should have the same format and structure as the /etc/apt/sources.list file does.

Another way of adding repositories to be searched is by creating a new text file (ending in .list) for it in the directory /etc/apt/sources.list.d/. This is a cleaner and more organized way of keeping track of your repositories, and you can always remove the repository by deleting that file.

As we detailed in Section W26.7.3, editing these files from the command line allows a user to add, remove, or temporarily disable software repositories.

You can produce a backup of the configuration file sources.list before you edit it, using the following command:

$ **sudo cp /etc/apt/sources.list /etc/apt/sources.list.backup**

W26.7.3.2 Repository Listing Format in /etc/apt/sources.list

A typical repository line in either of these two files will look similar to the following:

deb http://us.archive.ubuntu.com/ubuntu/ xenial main

where:

deb (or deb-src), which refers to where the apt command will find binary packages (deb) or source packages (deb-src),

the actual URL, which apt will use in order to “pull” from the repository,

the codename of the release; in this case, xenial (which refers to the codename for Ubuntu 16.04, "Xenial Xerus"), and

the Component named main, which references whether or not the repository contains software that is free and open source, and is supported officially by Canonical (the company that oversees Ubuntu's development). The component can be one of five of the following for Debain or Ubuntu- main, restricted, universe, or multiverse. For Linux Mint, there are five main components: main, upstream, import, backports, and romeo. To see a listing of all packages in the Linux Mint repository, go to- http://packages.linuxmint.com

Software packages have source code available, so if you are a developer, you are able to use this repository to fix bugs by altering the source code, and “packaging” up your changes. Software marked restricted is still supported, but may have a questionable license. Universe packages are supported by the community, not the maintainers of the software. Multiverse packages contain software that is not free and not supported.

In-Chapter Exercise W26.15

Name the parts of the entries for the following two lines in a /etc/apt/sources.list file:

deb http://download.webmin.com/download/repository sarge contrib

deb http://webmin.mirror.somersettechsolutions.co.uk/repository sarge contrib

W26.7.3.3 Other Suggested Procedures for Repository Management

1. Always back up configuration files like /etc/apt/sources.list before you begin editing. You can then reinstate your original source listings if something goes wrong with the entries in the newly added repository.

2. If you decide to add other repositories to sources.list, make sure that the repository actually works with your release of the software. Repositories that are not designed to work with your version can introduce faults in your operating system, and might force you to re-install the entire system!

3. Also, make sure that you really need to add external repositories as the software package(s) you are looking for may already have been introduced into the official repositories.

4. You may be asked enter a security key when adding a non-Ubuntu repository to your sources, as we did in the beginning of Section W26.7.3 with Webmin.

5. As much as possible, retrieve updated package lists by using the sudo apt-get update command when you have edited /etc/apt/sources.list.

W26.8 Tasks, Processes, Threads, and Traditional Process Control/Monitoring

Before discussing traditional methods of process control, and how to monitor processes, it is instructive at this point to know what a “task” refers to in Linux. In particular, knowing what the difference is, if any, between a process and a thread, in the context of Linux tasks. And this is useful knowledge, especially with respect to the Linux kernel itself. To make clear the differences between tasks, processes and threads, consider this set of comparisons between them-

A preliminary sketch of what a Linux “task” is, relative to processes and threads, and with reference to the kernel, is as follows:

In Linux, a task and a thread are synonymous, on a conceptual level.

A process at this level can be thought of as a completely independent virtual environment, which runs at least one task. A very good example is a single-thread process.

Each task is an independently-executing module within the virtual environment of a process.

The main task of a process (usually called the “leading thread”), defines the Task ID (TID) number, which is the same as the nominal Process ID (PID) number.

Every new thread that is spawned by a process, using the system calls that we detail extensively in Chapter 16, creates a new task within the process.

In order to identify these new tasks in the kernel, they get assigned their own individual Task ID (TID) number.

All tasks within a process share the same Thread Group ID (TGID) number.

A more complete description of what the difference is between processes and threads is given by the following listing of points:

1. With respect to the Linux kernel, a task that can be run and scheduled through the CPU is called a "process". We have described kernel scheduling through the CPU in Sections 10.2, and Section 10.5.1.

2. Each process has a globally unique Process ID (PID) number, as detailed in Section 10.5 , and a Thread Group ID (TGID) number.

3. A “stand alone” process, or “single-threaded” process, has a PID that is equal to its TGID, and no other process can have this TGID value.

4. A "threaded" process has its TGID value shared by other processes: they all have the same TGID value.

5. Processes sharing the same TGID also share the same memory space, signal handlers, etc.., as detailed in several sections of Chapter 16.

6. If a "threaded" process has a PID that is equal to its TGID, it can be called "the leading thread".

In a systems program, making the system call **getpid()** from within a process, will return its TGID (or "leading thread" PID).

7. From within a systems program, making the system call **gettid()** from within a process will return its PID.

8. Both a process and thread can be created with the very important Linux-specific **clone()** system call.

9. “Numbered” folders in /proc, that you can list with the ls command, are TGIDs.

10. “Numbered” folders in /proc/TGID/task contain numbers that are PIDs.

11. Even though you don't see every existing PID using the command ls /proc, you can still give the command cd /proc/any\_PID, where any\_PID is the PID of a process of interest.

12. In conclusion, and with respect to the kernel, only processes exist. Each process has its own PID. A thread is just a different kind of process.

Questions: “Are all processes threads, and why?”, and “Is the Linux kernel itself a process, and what are the threads created by this process?”

Answers: Since Linux is a multi-tasking, concurrent execution environment, all processes can be considered threads of execution through the system.

And at the conceptual level we are addressing here, the Linux kernel is not a process, but can be thought of as simply a complex interrupt handler, that controls the movement of tasks as threads through the CPU, and the CPU’s supportive virtualized memory system and persistent media.

For more information on threads, and the similarities and differences between processes and threads, see Chapter 16, ”System Programming II Process Management and Signal Processing”, particularly Sections 16.2 and 16.3.

W26.8.1 Controlling and Managing CPU Consumption by Processes

The traditional way of managing CPU consumption of a process is to use the nice and renice commands. In this sub-section, we illustrate the way you can use those commands effectively, after monitoring and assessing a processes activity.

As was stated explicitly in Chapter 10, Section 2, only root, or a user with superuser privileges, can increase a process CPU priority by decreasing its nice value.

In Chapter 17, Section 8.1.2, we have shown a more contemporary method of managing CPU priority with systemd cgroups.

The following code example shows you the display of the nine top current processes using the most CPU resources running on the system. When we type this in on our Linux Mint system, we get the output shown:

$ **ps -aux | head -10**

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

root 1 0.0 0.1 120360 6488 ? Ss Oct20 0:17 /lib/systemd/systemd --system --deserialize 32

root 2 0.0 0.0 0 0 ? S Oct20 0:00 [kthreadd]

root 3 0.0 0.0 0 0 ? S Oct20 0:08 [ksoftirqd/0]

root 5 0.0 0.0 0 0 ? S< Oct20 0:00 [kworker/0:0H]

root 7 0.0 0.0 0 0 ? S Oct20 8:05 [rcu\_sched]

root 8 0.0 0.0 0 0 ? S Oct20 0:00 [rcu\_bh]

root 9 0.0 0.0 0 0 ? S Oct20 0:00 [migration/0]

root 10 0.0 0.0 0 0 ? S Oct20 0:03 [watchdog/0]

root 11 0.0 0.0 0 0 ? S Oct20 0:03 [watchdog/1]

$

The pgrep command displays the PIDs of running processes. Here are some examples of how to use pgrep to find the process IDs of the running processes and pipe those PIDs to another command to produce the output.

To search for kthreadd and run ps (assumes you are running the Linux Mint Cinnamon desktop environment)-

$ **ps -p `pgrep kthreadd`**

PID TTY TIME CMD

2 ? 00:00:00 kthreadd

$

To search for systemd and run ps-

$ **ps -fp `pgrep systemd`**

PID TT STAT TIME COMMAND

UID PID PPID C STIME TTY STAT TIME CMD

root 1 0 0 Oct20 ? Ss 0:17 /lib/systemd/systemd --system --deserialize 32

root 341 1 0 Oct20 ? Ss 0:02 /lib/systemd/systemd-journald

root 1657 1 0 Oct20 ? Ss 0:01 /lib/systemd/systemd-logind

bob 2232 1 0 Oct20 ? Ss 0:00 /lib/systemd/systemd --user

root 17091 1 0 Oct23 ? Ss 0:00 /lib/systemd/systemd-udevd

$

To search for nginx , and improve its priority access to the CPU (assumes you have the nginx daemon installed and running)-

$ **sudo renice -1 `pgrep nginx`**

[sudo] password for bob: QQQ

7630 (process ID) old priority 0, new priority -1

7631 (process ID) old priority 0, new priority -1

7632 (process ID) old priority 0, new priority -1

$

To change nginx back to priority 0-

$ **sudo renice 0 `pgrep nginx`**

7630 (process ID) old priority 1, new priority 0

7631 (process ID) old priority 1, new priority 0

7632 (process ID) old priority 1, new priority 0

The nice and renice commands, as seen in the previous examples, change process priorities in the CPU.

Here is an example of using a command with nice to change a command’s nice value-

Run the vi text editor at a higher priority-

$ **sudo nice -n -1 vi**

When a process is already running, you can change the process’s nice value using the renice command. Here are some examples of the renice command-

Renice sarwar’s processes +2:

$ **renice +2 -u sarwar**

Renice PID 2576 by +5

$ **renice +5 2576**

Renice sarwar’s ksmserver processes to –3:

$ **renice -3 `pgrep -u sarwar ksmserver `**

2545: old priority -1, new priority -3

2546: old priority -1, new priority -3

2547: old priority -1, new priority -3

The back quotes are used in the previous command lines to show that the output of the pgrep command (usually PIDs ) should be redirected to the either the nice or renice command. The nice settings for your processes are displayed by default (as NI in the display below) when you run the top command, as shown in more detail in Chapter 10, Section 10.5.2. You can run the top command, with a delay of screen updates every 10 seconds, as follows:

$ **top -d 10**

top - 06:00:36 up 1 day, 15:40, 1 user, load average: 0.12, 0.20, 0.19

Tasks: 258 total, 1 running, 257 sleeping, 0 stopped, 0 zombie

%Cpu(s): 0.5 us, 0.4 sy, 0.0 ni, 99.1 id, 0.1 wa, 0.0 hi, 0.0 si, 0.0 st

KiB Mem : 5981784 total, 3483416 free, 749348 used, 1749020 buff/cache

KiB Swap: 6157308 total, 6157308 free, 0 used. 4876824 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND 15159 root 20 0 0 0 0 S 0.5 0.0 0:14.54 kworker/0:0

2228 root 20 0 399344 92040 55144 S 0.4 1.5 25:22.85 Xorg

2570 bob 20 0 1572696 193004 69664 S 0.3 3.2 31:44.13 cinnamon

7 root 20 0 0 0 0 S 0.2 0.0 3:35.24 rcu\_sched

15079 bob 20 0 485236 36100 27676 S 0.2 0.6 0:03.94 gnome-term+

1 root 20 0 185604 6160 3896 S 0.0 0.1 0:07.46 systemd

Output truncated...

Based upon the output of the top command, you could stop an unimportant or runaway process with the kill command, as follows. We previously had determined the PID of an unimportant process as 19993. Be careful here, if you kill an important process, your system goes down!

$ **sudo kill -9 19993**

$

W26.8.2 Disk Usage and Management

According to our recommended storage model, our Linux system is on a single independent disk, and the data disk is a separate file-posing-as-device, i.e. a “simulated” vdevs in the ZFS storage pool named lxd\_pool. For a more detailed description of this mechanism, see Chapter W22 on ZFS and Chapter W23 on Virtualization Methodologies at the book website dealing with “containers” using LXC/LXD.

The installation of ZFS on our Debian-family and CentOS systems is detailed in Appendix A of the printed book.

zpool list: Lists existing or all or indicated pools, along with health status and space usage

zpool status: Displays the detailed health status of all or indicated pools

zpool iostat: Displays I/O statistics for all or indicated pools

zpool upgrade -a: Upgrades all pools to the latest available version

zpool set deduplication on/off: Sets the deduplication property of a pool

zpool scrub: Examines all data in the named pool and verifies that the checksums are correct

zfs upgrade: Upgrades file systems to a new on-disk version

du: Displays the block usage for file systems and directories

For ZFS, Linux Mint is currently at version 5.

The following set of commands show the output of the preceding commands executed on a Linux Mint system.

$ **sudo zpool list**

[sudo] password for bob: **QQQ**

NAME SIZE ALLOC FREE EXPANDSZ FRAG CAP DEDUP HEALTH ALTROOT

lxd\_pool 1008M 858M 150M - 62% 85% 1.00x ONLINE -

$ **sudo zpool list lxd\_pool**

NAME SIZE ALLOC FREE EXPANDSZ FRAG CAP DEDUP HEALTH ALTROOT

lxd\_pool 1008M 858M 150M - 62% 85% 1.00x ONLINE -

$ **sudo zpool iostat**

capacity operations bandwidth

pool alloc free read write read write

---------- ----- ----- ----- ----- ----- -----

lxd\_pool 858M 150M 0 101 5.03K 227K

$ sudo zpool iostat lxd\_pool

capacity operations bandwidth

pool alloc free read write read write

---------- ----- ----- ----- ----- ----- -----

lxd\_pool 858M 150M 0 101 5.03K 227K

$ **sudo zpool status**

pool: lxd\_pool

state: ONLINE

scan: none requested

config:

NAME STATE READ WRITE CKSUM

lxd\_pool ONLINE 0 0 0

/home/bob/disk\_file ONLINE 0 0 0

errors: No known data errors

$ **sudo zfs list**

NAME USED AVAIL REFER MOUNTPOINT

lxd\_pool 858M 118M 19K none

lxd\_pool/containers 190M 118M 19K none

lxd\_pool/containers/container1 190M 118M 738M /var/lib/lxd/containers/container1.zfs

lxd\_pool/images 667M 118M 19K none

lxd\_pool/images/d0c…a2c3b 667M 118M 667M /var/lib/lxd/images/d0c…a2c3b.zfs

$ **sudo zpool upgrade**

This system supports ZFS pool feature flags.

All pools are formatted using feature flags.

Every feature flags pool has all supported features enabled.

$ **sudo zfs upgrade**

This system is currently running ZFS filesystem version 5.

All filesystems are formatted with the current version.

$ **sudo du > dufile**

$more dufile

4 ./Videos

4 ./.local/share/Trash/files

4 ./.local/share/Trash/info

4 ./.local/share/Trash/expunged

16 ./.local/share/Trash

4 ./.local/share/icc

12 ./.local/share/keyrings

4 ./.local/share/cinnamon/applets

4 ./.local/share/cinnamon/extensions

4 ./.local/share/cinnamon/search\_providers

4 ./.local/share/cinnamon/desklets

20 ./.local/share/cinnamon

4 ./.local/share/nemo/scripts

4 ./.local/share/nemo/actions

12 ./.local/share/nemo

... [Output truncated]

W26.8.4 systemd journal Log Messages

Question: What do we use the systemd Journal for primarily?

Answer: To check for unauthorized entry into our system, and to monitor system performance.

But it has many other uses, for example in monitoring system performance.

The journal is created and controlled by the journald daemon, which directs all of the messages produced by the kernel, initrd, services, etc. into a binary record structure. The systemd journal is a single, centralized management tool for logs, regardless of where the log messages are sent from.

One salient feature of using systemd is that log messages currently can be output using the traditional message printing APIs with the syslog function call, as well as by using the journal API function calls.

And a critical, and somewhat controversial aspect of systemd journal logging, is that the log files are stored as binary data, and can be searched by specialized database traversal techniques.

Storing the log data in a binary format most importantly means that the data can be displayed in useful output formats.

Access to the logs kept by the journal daemon is done using the journalctl command. The chapter on systemd at the Github site fully details the use of the journalctl command and its options and arguments. In this section, we give you a basic overview of what can be viewed with the journalctl command.

W26.8.4.1 journalctl Basics

A very simple first command to use when you want to view logs with the journalctl command, is to type that command with now options or arguments: To have a first look at the logs, just type in:

$ **journalctl**

The output you get is very similar to the traditional output when viewing system logs. With the following notable exceptions:

Lines of error priority (and higher) will be highlighted red.

Lines of notice/warning priority will be highlighted bold.

The timestamps are converted into your local time-zone.

The output is auto-paged by pressing the space bar.

This will show all available data, including rotated logs.

Between the output of each boot we'll add a line clarifying that a new boot begins now.

By default, ordinary, unpriveleged users can only watch their own logs, but if that user is in the administrative group. To add an ordinary, unprivileged user to the adm group, use the following command:

$ **sudo usermod -a -G adm hassan**

After logging out and back in as hassan, have access to the full journal of the system and all users:

To view logs as they grow, use the following command:

$ **journalctl -f**

This command shows the last ten Journal log lines, and then waits for changes and show them as they take place.

To view the Journal logs of just the current boot environment (since the last reboot), use the following command:

$ **journalctl -b**

Listing all log messages with priority levels ERROR and worse, from the current boot environment, use the following command:

$ **journalctl -b -p err**

To see Journal entries in a more restricted timeframe, for example from yesterday until now, type the following command:

$ **journalctl --since=yesterday**

All log messages from the day before at 00:00 in the morning until right now are shown on screen.

To search for entries that recorded in the journal on the the 25th of October, until today, use the following command:

$ **journalctl --since=2017-10-25 --until=today**

To see messages logged to the journal by a particular service unit, such as nginx, use the following command:

$ **journalctl -u httpd --since=00:00 --until=9:30**

Finally, to see journal entries for a particular device, such as the disk drive at /dev/sda, use the following command:

$ **journalctl /dev/sda**

W26.9.2 Access Control Credentials: Discretionary (DAC), Mandatory (MAC), and Role-Based (RBAC)

The nomenclature we use in this section deserves some attention. When we talk about access control via security checks, here are some important terms-

Objects: The entities that are targeted, or worked on, by the processes of a program. For example, processes themselves can be objects, or the processes generated by executing instances of a program.

Files/inodes are another form of object, particularly the executable form of file objects, and the attendant data structure holding their information. Not to be confused with file system objects, which we have so far referred to as either an ordinary file or a directory.

Object Ownership: Indicates the owning user and group.

Object Context: Security checks done when objects are acted on.

Subjects: An object that is acted upon by another object. Processes are active subjects, such as those processes that are created by an **exec()** or **fork()** system call from an originating process.

Subject Context: Security checks done when an active subject performs its operations.

Action: What a subject does to an object. This includes reading, writing, creating and deleting files; forking or signaling.

Permissions: Security checks when a subject acts upon an object. Taking the subject context, the object context, and the action, and searching one or more sets of permissions to see whether the subject is granted or denied permission to act in the desired manner on the object, given those contexts. In simple terms, match subject and object permissions, and let the subject act or not on the object.

There are three basic “classes” of permissions-

1. Discretionary Access Control (DAC):

Sometimes the object will include sets of rules as part of its description. This is an 'Access Control List' or 'ACL'. A Linux file may supply more than one ACL. A traditional Linux file, for example, includes a permissions mask that is an abbreviated ACL with three fixed classes of subject ('user', 'group' and 'other'), each of which may be granted certain privileges ('read', 'write' and 'execute' - whatever those map to for the object in question). Linux file permissions do not allow the arbitrary

specification of subjects, however, and so are of limited use.

A Linux file might also support a POSIX1e ACL, or in the case of ZFS, an NFSv4 ACL. This is a list of rules that grants various permissions to arbitrary subjects.

2. Mandatory access control (MAC):

The system as a whole may have one or more sets of permissions that get applied to all subjects and objects, regardless of their source. Security Extended Linux (SELinux) is an example of this.

3. Role-Based Access Control:

Rather than use the user ID to determine what access rights users and groups have on the system, the Role-Based Access Control (RBAC) model grants access based on the role or roles that a user assumes. The classic RBAC example is the use of the su and sudo commands to grant an unprivileged user root privileges. Another example xan be found in ZFS, when you execute the **zfs** command, and your action is checked to see that the subject issuing the command has the role privilege, even if the user is root.

These forms of access control policies determine what access action is allowed on what object, under what circumstances (DAC, MAC, or Role-Based Access Control (RBAC)) and by what subject.

A permission, for example, can be thought of as read, write, or execute privilege. A subject, for example, can be thought of as an executing process. Most importantly, an object is a Linux process (detailed in Chapter 10), since everything done on files and the data in them, is done through active processes.

On the command line, an ordinary user or the system administrator is able to implement resource use restrictions and privileges by controlling process credential assignments, exercised on subject executable image files, via the **chmod** command. On Linux, an ordinary unprivileged user, can be given the required privileged role with the **su** or **sudo** command. Then, as root, she can issue a privileged **chmod**, **chown**, and **chgrp** to grant or modify file and directory access permissions, and use the DAC, MAC, or RBAC methods.

Types of Credentials

We are concerned with the three basic types of credentials that the Linux kernel supports. These are as follows, with major references in the text shown:

\* Traditional UNIX Credentials – Chapter 5

1. Real User ID

2. Real Group ID

UID and GID are assigned to most Linux objects. These in large part define the object context of that object, with processes included in this assignment.

3. Effective (EUID), Saved (SUID) and FS (FSID) User ID

4. Effective (EGID), Saved (SGID) and FS (FSGID) Group ID

5. Supplementary groups

The additional credentials used by processes are EUID/EGID/GROUPS, and are used as the subject context, and real UID/GID will be used as the object context.

W26.9.2.1 sudo

The sudo program allows a single command to be run as root, or even as some other user. The system administrator utilizes a policy listing file (named sudoers) that contains commands that each user can execute. When any user needs to run a command that requires root permissions, that user types sudo command in a console terminal, allowing them to run command. Then, sudo consults its permissions list in the policy listing file. If the user has permission to run that command, it runs the command. If the user does not have permission to run the command, sudo denies execution. Running sudo does not require knowing root’s password, but by default requires the user’s own password to execute successfully.

An important security access consideration, before you allow someone to execute sudo, is finding out what group or groups you, or any other user for that matter, actually belong to on your system. The user you want to include in the sudoers file may already belong to a group that has adequate access privileges! For you, this can be simply done by executing the following command:

$ **id**

uid=1000(bob) gid=1000(bob) groups=1000(bob),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(netdev),113(lpadmin),130(sambashare),132(lxd)

From the above output on our system, user bob belongs to the groups bob, adm, cdrom, sudo, dip, plugdev, netdev, lpadmin, sambashare, and lxd. Therefore, he already has access to files and directories that those groups have access to.

To find out what groups are defined on the system, you can use the following commands:

$ **compgen -g**

root

daemon

bin

sys

adm

tty

disk

Output truncated...

In this way, before assigning groups or users in sudoer file “aliases”, you have an idea of what groups exist on the system. We detail sudoer aliases in the following sections.

There are two aspects to sudo: the sudo program itself, and the sudoers policy file that the program uses. The sudoers policy file can only be edited by root. The sudo program includes a special tool, visudo, just for editing and validating the sudoers policy file. The path to the executable program visudo is /usr/sbin.

The sudoers policy file must only be edited with visudo, because that special editing tool has safeguards built into it. Typing visudo launches the vi text editor to allow editing of the sudoers file. The sudoers file itself is found in /etc. At this point, you should use the sudo more command to examine the contents of the sudoers file on your system.

The sudoers file recognizes seven types of user specification lists. They are usernames, group names (such as lxd), aliases defined within the sudoers file itself, UID numbers, GID numbers, netgroups, and non-UNIX groups. See the following section for applications of some of these seven specification lists.

The sudoers file is composed of two types of entries: aliases, and user specifications. Aliases are names that can be assigned to a variety of groups of objects, like users, hosts, etc..

User specifications dictate who may run what. When multiple entries match a user, they are applied in order. Where there are multiple matches, the last match is used (which is not necessarily the most specific match). A user specification determines which commands a user may run (and as what user) on specific hosts.

By default, commands are run as root, but this can be changed on a per-command basis.

In the following two sections, we give examples of alias entries and various forms of user specification.

W26.9.2.1.1 Alias Specifications and Definition in the Sudoers File

There are four kinds of aliases: User\_Alias, Runas\_Alias, Host\_Alias and Cmnd\_Alias.

A generalized syntactic description of these alias specifications is as follows:

Alias = 'User\_Alias' User\_Alias (':' User\_Alias)\* |

'Runas\_Alias' Runas\_Alias (':' Runas\_Alias)\* |

'Host\_Alias' Host\_Alias (':' Host\_Alias)\* |

'Cmnd\_Alias' Cmnd\_Alias (':' Cmnd\_Alias)\*

User\_Alias = NAME '=' User\_List

Runas\_Alias = NAME '=' Runas\_List

Host\_Alias = NAME '=' Host\_List

Cmnd\_Alias = NAME '=' Cmnd\_List

NAME = [A-Z]([A-Z][0-9]\_)\*

An alias definition is given in the following format:

Alias\_Type NAME = object1, object2, ...

where:

Alias\_Type is one of User\_Alias, Runas\_Alias, Host\_Alias, or Cmnd\_Alias.

NAME is a string of uppercase letters, numbers, and underscore characters (‘\_’).

NAME must start with an uppercase letter. To put several alias definitions of the same type on a single line, joined by a colon (‘:’). For example-

Alias\_Type NAME = object1, object2... : NAME = object4, object5...

You cannot redefine an existing alias. But, it is possible to use the same name for aliases of different types, but a name collision is possible which would generate an error.

The following is a typical alias entry in a sudoers file, where group1 can be an alias name that includes more than one user. This alias specifies that the user group1 may run /bin/ps, /bin/nano, and /usr/sbin/vsftpd , but only as the user admin on the host named ProLiant-MicroServer-

group1 ProLiant-MicroServer = (admin) /bin/ps, /bin/nano, /usr/bin/vsftpd

W26.9.2.1.2 User Specifications in the Sudoers File

The user specifications in the sudoers file contain policy rules, one rule per line. Every rule uses the general format as follows:

*who where = (as\_whom) what*

where:

*who* is the user that this rule applies to. who can also be a user specification list—for example, a group name , as shown in Section W26.3.2, “Adding/Deleting and Maintaining Users and Groups ”

*where* is the hostname of the system this rule applies to.

*=* separates the where from (as\_whom) and what.

*(as\_whom)* designates the user specification list sudo will run the what.

*what* lists the full path to each command this policy rule applies to.

You must specify full pathnames to command(s) you include in the sudoers file.

A syntactic description of the kinds of user specifications is as follows:

User specification

User\_Spec = User\_List Host\_List '=' Cmnd\_Spec\_List \

(':' Host\_List '=' Cmnd\_Spec\_List)\*

Cmnd\_Spec\_List = Cmnd\_Spec |

Cmnd\_Spec ',' Cmnd\_Spec\_List

Cmnd\_Spec = Runas\_Spec SELinux\_Spec Tag\_Spec\* Cmnd

Runas\_Spec = '(' Runas\_List (':' Runas\_List)? ')'

SELinux\_Spec := ('ROLE=role' | 'TYPE=type')

Tag\_Spec = ('EXEC:' | 'NOEXEC:' | 'FOLLOW:' | 'NOFOLLOW' |

'LOG\_INPUT:' | 'NOLOG\_INPUT:' | 'LOG\_OUTPUT:' |

'NOLOG\_OUTPUT:' | 'MAIL:' | 'NOMAIL:' | 'PASSWD:' |

'NOPASSWD:' | 'SETENV:' | 'NOSETENV:')

The following command uses the -l option to sudo to allow you to list the permissions currently defined as policy in the sudoers file for the user typing in the command.

$ **sudo -l**

Matching Defaults entries for bob on bob-ProLiant-MicroServer:

env\_reset, mail\_badpass,

secure\_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User bob may run the following commands on bob-ProLiant-MicroServer:

(ALL : ALL) ALL

(root) NOPASSWD: /usr/lib/linuxmint/mintUpdate/checkAPT.py

We see from the above output, for example, that user bob can run all commands as root.

In the following simple user specification rule, mansoor has the permission to run any command:

mansoor ALL = ALL

The following rule allows user mansoor to run the visudo program:

mansoor ALL = /usr/sbin/visudo

Using a group name in a sudoers alias entry has the following syntax, where everyone in the group named lxd, on the system named Proliant-Microserver, can run all of the commands in /etc/sbin as the user lxd:

$ **lxd Proliant-Microserver = (lxd) /etc/sbin\***

Notice that the group name is prefaced with the percent sign character (%).

Using a user ID number in a sudoers alias entry has the following syntax, where the user with ID 1002 can run everything in the /usr/sbin directory:

#1002 ALL = /usr/sbin/\*

Notice that the user ID number is prefaced with the pound sign character (#).

W26.9.2.1.3 An Example sudoers File

Following is a more complete sudoers file content listing. It is basically divided into two sections, the alias specifications first, and then the user specifications-

#Alias Specifications Section

# User alias specification

User\_Alias FULLTIMERS = mansoor, hassan, ibrahim

User\_Alias PARTTIMERS = mahan, jill, bob

User\_Alias WEBMASTERS = mansoor, bob, hassan

# Runas alias specification

Runas\_Alias OP = root, operator

Runas\_Alias DB = oracle, mariadb

Runas\_Alias ADMINGRP = adm, oper

# Host alias specification

Host\_Alias Proliant-Microserver = black, blue, red :\

Black-Dragon = time, space, architecture :\

Big-Blue = green, magenta :\

Linux\_Mint\_mini = fast, lang, python

Host\_Alias CUNETS = 192.168.0.8/255.255.0.0

Host\_Alias CSNETS = 192.168.0.6, 192.168.0.0/24

Host\_Alias SERVERS = master, mail, www, ns

Host\_Alias CDROM = sony, spintec

# Cmnd alias specification

Cmnd\_Alias KILL = /bin/kill

Cmnd\_Alias PRINTING = /usr/sbin/lpc, /usr/bin/lprm

Cmnd\_Alias SHUTDOWN = /usr/sbin/shutdown

Cmnd\_Alias HALT = /sbin/halt

Cmnd\_Alias REBOOT = /sbin/reboot

Cmnd\_Alias SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh,\

/usr/local/bin/tcsh, /usr/bin/rsh,\

/usr/local/bin/zsh

Cmnd\_Alias SU = /bin/su

Cmnd\_Alias PAGE = /bin/more, /bin/less

# User Specification Section

root ALL = (ALL) ALL

bob ALL = (ALL) ALL

FULLTIMERS ALL = NOPASSWD: ALL

PARTTIMERS ALL = ALL

#1

mansoor CSNETS = ALL

#2

maham CUNETS = ALL

#3

hassan ALL = /usr/bin/su operator

#4

%operators ALL = (: ADMINGRP) /usr/sbin/

#5

ibrahim Proliant-Microserver = (OP) ALL : Black-Dragon = (OP) ALL

#6

jessica ALL, !SERVERS = ALL

#7

cody SERVERS = /usr/bin/, !SU, !SHELLS

#8

ALL CDROM = NOPASSWD: /bin/umount /CDROM,\

/bin/mount -o nosuid\,nodev /dev/cd0a /CDROM

W26.9.3 Linux Access Control Lists (ACL’s)

ACLs provide the ordinary, unprivileged user with the ability to set finer access controls on directories and files than the traditional Linux permissions, whether they are used on EXT4, or ZFS filesystems. Two different basic types of ACL apply to files and directories. An ACL that defines the current access permissions of files and directories is called an *access* ACL. An ACL, which can only logically be set on a directory, and that defines the permissions that a directory object inherits from its parent directory at the time of its creation, is called a *default* ACL. Additional basic types of ACL are *minimal* and *extended* ACLs. ACL permissions that that can be equivalent to the traditional file mode permissions are called *minimal ACLs*. Minimal ACLs have three entries, which can be the same as the traditional file permissions. ACLs with more than the three entries are called extended ACLs. Extended ACLs also contain a mask entry and may contain any number of named user and named group entries.

For a more complete discussion of Linux Capabilities, see the man page for capabilities on your system. The scope of capabilities generally follows this outline-

1. Set of permitted capabilities

2. Set of inheritable capabilities

3. Set of effective capabilities

4. Capability bounding set

These make the most sense when they apply to processes, which are the active element in system operation. They are privileged permissions exercised in a “finer-grained” context. Finer-grained is used here to mean a more specific, targeted privilege, such as those shown in Table W26.14. These are applied to a process or processes that ordinarily, via the traditional model, could only be granted a blanket, all-or-none tableau of privileges. Including a user in the sudoers file is an example of this traditional models application.

Capabilities are controlled by changes in the traditional Linux permissions, but can also be set more finely and viewed directly by the capset and getcap system calls.

Using Access Control Lists (ACLs) in Linux

The traditional Linux permissions model, which defines secure access to file objects like a regular file or a directory, set permissions of read, write, and execute. We show that traditional permissions model in Chapter 5, Section 5.4. Other advanced techniques for setting permissions, as we have shown, are setting the setuid, setgid, and sticky bit. Beyond these, the Access Control List (ACL) model gives users finer-grained control over file and directory object security. Every file object can be thought of as having associated with it an access ACL that controls the discretionary access to that object. For a directory, this ACL is referred to as a default ACL.

Question: Why would an ordinary user, or a system administrator, want to use ACLs rather than just rely on the traditional permissions model?

Answer: To set discretionary controls on a file so that more than one group can access it. In the traditional permissions model, any file can only belong to one group. Therefore to serve different collections of users, many different groups have to exist. Only administrators can create and assign group membership. But with ACL’s, a file can belong to many groups, and in addition, can be given access by an ordinary user.

In-Chapter Exercise W26.16

Does it make any sense to set the permissions on a regular file, so that a group, user, or others have access to it, when that group, user, or others do not have permissions set to access the directory which contains the regular file?

If you have answered the question posed in the In-Chapter Exercise, you can see why the Examples presented in this section apply mainly to directories. ACLs provide the ordinary, unprivileged user with the ability to set finer access controls on directories and files, but at some significant performance costs, whether they are used on EXT4, or even possible on ZFS filesystems. Our Linux base system, Linux Mint, supports the traditional permissions model and the ACL model (referred to here as POSIX.1e). It is worth noting here that UNIX systems with ZFS only supports a style of ACLs known as NFSv4 ACLs. We show an extended example of the NFSV4 model.

We cover the basics of the following topics in the sub-sections indicated:

W26.9.3.1 Linux POSIX.1e ACL Model Details

W26.9.3.2 Command Syntax for Setting and Viewing ACLs

W26.9.3.3 Examples of setting ACL’s

W26.9.3.4 Setting NFSV4 ACLs on ZFS

In order to get more help, and to get more detail on ACLs on your system, we encourage you to consult the man pages for acl, setfacl, and getfacl.

W26.9.3.1 Linux POSIX.1e ACL Model Details

The Linux default implementation of ACLs uses POSIX.1e syntax. These ACLs are set and displayed with the **setfacl** and **getfacl** commands.

The traditional file system object permission model, that we illustrate in Chapter 5, defines three classes of users: owner, group, and other. Each of these classes is associated with a set of permissions. The permissions defined are read (r), write (w), and execute (x). In this model, the owner class permissions define the access privileges of the file owner, the group class permissions define the access privileges of the owning group, and the other class permissions define the access privileges of all users that are not in one of the previous two classes. The ls -l command displays the owner, group, and other class permissions as its output output. For example, rwxrw-r- - for some regular file, which translates to read, write, and execute permission for the owner class, read and write permission for the group class, and read permission for others.

W26.9.3.1.1 Basic Types of the POSIX.1e ACL Model

For our purposes here in this section, a “file system object” can be both a regular file or a directory. Two different basic types of ACL apply to files and directories. An ACL that defines the current access permissions of files and directories is called an *access* ACL. An ACL, which can only logically be set on a directory, and that defines the permissions that a directory object inherits from its parent directory at the time of its creation is called a *default* ACL. We give two extended examples of setting and viewing these two basic types of ACL in Section W26.9.3.3.

In-Chapter Exercise W26.17

Why is it logical that only directories can be associated with default ACLs?

Additional basic types of ACL are *minimal* and *extended* ACLs. ACL permissions that that can be equivalent to the traditional file mode permissions are called *minimal ACLs*. Minimal ACLs have three entries, which can be the same as the traditional file permissions. An example of a minimal ACL, obtained on the command line with the getfacl command, is as follows:

$ **getfacl acltest**

# file: acltest

# owner: bob

# group: bob

user::rwx

group::rwx

other::---

Note that the last three lines of output are the same as traditional permissions.

ACLs with more than the three entries are called extended ACLs. Extended ACLs also contain a mask entry and may contain any number of named user and named group entries. An example of the extended ACL type, obtained on the command line with the getfacl command, is as follows:

$ **getfacl acltest**

# file: acltest

# owner: bob

# group: bob

user::rwx

user:mansoor:rwx

group::rwx

mask::rwx

other::---

W26.9.3.1.2 How Permissions Map to the Basic ACL Types

An ACL, whether it is access, default, simple, or extended as defined above, draws its permission classes from those shown in Table W26.7. Each of the three traditional file permission classes (owner, group, others) is represented by an ACL entry. The only other classes that are added in an extended ACL are permissions for additional users or additional groups. This is very critical to understanding POSIX.1e ACLs, because this feature allows a file object to have its permissions controlled by and associated with different users and groups- the very objective gained by using POSIX ACLs in the first place.

Table W26.7 shows the allowed Entry Types and their text formats, as opposed to their numerical forms. Each Text Form in the table consists of a type, a qualifier that specifies to which user or group the entry applies, and a set of permissions.

Table W26.8 shows ACL entries and their descriptions.

Table W26.9 shows default ACL entries and their descriptions.

|  |  |
| --- | --- |
| Entry Type | Text Format |
| Owner | user::rwx |
| Named user | user:name:rwx |
| Owning group | group::rwx |
| Named group | group:name:rwx |
| Mask | mask::rwx |
| Others | other::rwx |

Table W26.7: Allowed Types of POSIX.1e ACL Entries

| ACL Entry | Description |
| --- | --- |
| u[ser]::perms | File owner permissions. |
| g[roup]::perms | File group permissions. |
| o[ther]:perms | Permissions for users other than the file owner or members of file group. |
| m[ask]:perms | The ACL mask. The mask entry indicates the maximum permissions allowed for users (other than the owner) and for groups. The mask is a quick way to change permissions on all the users and groups.  For example, the mask:r-- mask entry indicates that users and groups cannot have more than read permissions, even though they might have write/execute permissions. |
| u[ser]:uid:perms | Permissions for a specific user. For uid, you can specify either a user name or a numeric UID. |
| g[roup]:gid:perms | Permissions for a specific group. For gid, you can specify either a group name or a numeric GID. |

Table W26.8 ACL Entries in Extended BNF

| Default ACL Entry | Description |
| --- | --- |
| d[efault]:u[ser]::perms | Default file owner permissions. |
| d[efault]:g[roup]::perms | Default file group permissions. |
| d[efault]:o[ther]:perms | Default permissions for users other than the file owner or members of the file group. |
| d[efault]:m[ask]:perms | Default ACL mask. |
| d[efault]:u[ser]:uid:perms | Default permissions for a specific user. For uid, you can specify either a user name or a numeric UID. |
| d[efault]:g[roup]:gid:perms | Default permissions for a specific group. For gid, you can specify either a group name or a numeric GID. |

Table W26.9 ACL Default Entries in Extended BNF

In-Chapter Exercise W26.18

What undefined qualifiers do not require a specification of qualification, as seen in Table W26.7? Why?

W26.9.3.1.3 The Meaning of the “Mask” Entry

The extended classes, “Named group” and “Named user”, assign permissions to the group class, which already contains the owning group entry. In minimal ACLs, the group class permissions are the same as the owning group permissions. In an extended ACL, the group class contains ACL entries with different permission sets than the group class. That is the meaning of “extended”. The group class permissions can possibly contain conflicting permission sets, given that they now have several competing specifications included in them. This raises the possibility of an inconsistency. For example, the results of using the getfacl command on the directory file object named acltest, after adding an extended ACL specification of r—for the already-existing group named “development”, yields the following:

$ **getfacl acltest**

# file: acltest

# owner: bob

# group: bob

user::rwx

user:mansoor:rwx

group:development:r--

group::rwx

mask::rwx

other::---

As found in the example, with extended ACLs, the Owning group (bob) class permissions (rwx) are mapped to what is known as the “mask”, or masking entry, with its attendant permissions. The Owning group (bob) entries (rwx) still define the owning group permissions.

The group class permissions for the Owning group bob as shown (rwx) are more “inclusive”, that is, they include more permissions. They represent the widest, most inclusive set of the permissions that any entry in the Owning group class will grant. On the file object acltest, to handle the “inconsistency” of differences in Named group and Owning group permissions, the “masking” permissions are used, and are specified as shown in the example, as the mask entry. This means that permissions in entries that are a member of the group class are also present in the mask entry, and are effective.

Take note that permissions that are absent in the mask entry are masked and do not take effect.

In-Chapter Exercise W26.19

Construct a hypothetical extended ACL on paper that shows how a named user mansoor, with an rw- set of permissions sets the mask of the Owning group class, even though the Owning group permissions are set to r--.

So what do the mask entries actually mean and achieve? Following we show some specific explanatory examples of mask entry use. The first example shows what the extended ACLs would look like after making a change to the users traditional permissions, with the chmod command. Notice it has no affect on the ACL mask entries-

$ **chmod u-w acltest**

$ **getfacl acltest**

# file: acltest

# owner: bob

# group: bob

user::r-x

user:mansoor:rwx

group::rwx

group:development:r--

mask::rwx

other::---

When the chmod command changes any of the traditional permissions, for the owner, group, or other class, the ACL entry changes as well. And when the Owner class ACL entry permissions change, via use of the setfacl command and its option, the traditional permissions of the Owner class change also.

But if you change the group traditional permissions with chmod, notice the effect on the ACL mask entries -

$ **chmod g-w acltest**

$ **getfacl acltest**

# file: acltest

# owner: bob

# group: bob

user::r-x

user:mansoor:rwx #effective:r-x

group::rwx #effective:r-x

group:development:r--

mask::r-x

other::---

If an ACL entry contains permissions that are disabled by the mask entry, such as user mansoor’s write access, and the Owning groups write access, the getfacl display shows a comment next to each that signifies the “effective” set of permissions that are given by that entry.

After executing the chmod command to remove the write permission from the group class, the mask permissions have changed. Now write permission is limited to the owner of the directory acltest. This output includes a comment for all those entries in which the effective permissions do not correspond to the original permissions, because they are modified by the mask entry.

In-Chapter Exercise W26.20

How can you restore the original group write permission on the acltest directory?

In conclusion, with extended ACLs, masking permissions do the following : permissions in entries that are a member of the group class which are also in the mask entry are effective. Permissions that are not in the mask entry are masked and do not take effect.

The owner and other entries are not in the group class. Their permissions are always effective and not masked.

W26.9.3.1.4 Drawbacks and Alternatives to the POSIX.1e ACL Model

Some of the drawbacks of the POSIX.1e ACL model are as follows:

Unfortunately, at the time of the writing of this book, Linux and the POSIX.1e ACL model that it supports, does not easily allow for the interoperability, in terms of NSFV4 ACLs, between Linux and non-Linux servers and clients. For example, if a Linux user were to mount an NFSV4 ACL-compliant filesystem as a shared resource, the default POSIX.1e implementation on Linux would not be able to take advantage of the ACL model of that filesystem. In order to do so, a user would have to install a special tool, nfs4-acl-tools, in addition to enabling NFS server facilities on the remote system, and client facilities on her Linux system. Since that is necessarily a more advanced application of ACLs, we do not cover that particular situation in this chapter.

But we do give a brief introduction to NFSV4 ACLs, applied in a Virtual Machine environment running UNIX TrueOS, in anticipation of the adoption of NFSV4 ACLs in Linux. In preparation for that introduction, we recommend that you proceed with doing the ZFS and Virtualization Methodologies at the Github site to familiarize yourself with ZFS and Virtual Machines.

The TruesOS operating system uses ZFS for its root filesystem, and ZFS fully supports NFSV4 ACLs.

Therefore, to gain an appreciation now of what advantages NFSV4 ACLs give the ordinary user, above and beyond POSIX.1e ACLs, we show NSFV4 ACLs in TrueOS.

Another very practical limitation in this respect is shown in Chapter W22 on virtualization at the book website for LXD containers. In that chapter, we create the containers so that they by default use ZFS. So in the Linux host for those containers, the NSFV4 ACLs could not be used on LXD container file objects.

Additionally, ACLs are not retained by the tar command.

W26.9.3.2 Command Syntax for Setting and Viewing POSIX.1e ACLs

To set or modify ACLs, use the **setfacl** command. To see the results of using **setfacl**, use the **getfacl** command. Following are an abbreviated syntax description of those two commands, with common and allowable POSIX.1e ACL options shown-

**setfacl** - set file access control lists

**Syntax: setfacl [-bkndRLPvh] [{-m|-x} acl\_spec] [{-M|-X} acl\_file] file ...**

**setfacl --restore=file**

**Purpose:**

The **setfacl** utility sets discretionary access control information on the

specified file(s).

**Output:** Modified ACL specifications.

**Common Options and Option Arguments:**

-b Remove all ACL entries except for base ACL entries owner, group, others.

-m [entries]

Modify the ACL on the specified file. New entries will be added,

and existing entries will be modified according to the entries

argument. For NFSv4 ACLs, you can also use the -a and -x

options instead.

-M file

Modify the ACL entries on the specified files by adding new ACL

entries and modifying existing ACL entries with the ACL entries

specified in the file file.

-x entries

If entries is specified, remove the ACL entries specified there

from the access or default ACL of the specified files.

**getfacl -** get file access control lists

**Syntax: getfacl [-aceEsRLPtpndvh] file ...**

**getfacl [-aceEsRLPtpndvh] -**

**Purpose:**

The **getfacl** utility writes discretionary access control information associated with the specified file(s) to standard output

**Output:** Indicated file ACL settings on stdout.

The complete output format of an example getfacl command, and of which we show many example cases in the sub-sections below, is as follows:

1: # file: filename

2: # owner: bob

3: # group: admin

4: # flags: -s-

5: user::rwx

6: user:mansoor:rwx #effective:r-x

7: group::rwx #effective:r-x

8: group:developers:r-x

9: mask::r-x

10: other::r-x

11: default:user::rwx

12: default:user:mansoor:rwx #effective:r-x

13: default:group::r-x

14: default:mask::r-x

15: default:other::---

**Common Options:**

-a Display the file access control list.

-d Display the default access control list.

-n Display numeric user and group IDs .

W26.9.3.3 ACL Examples

The following Examples provide the ordinary user with the ability to set and view POSIX.1e ACL attributes on files and directories.

Example W26.21 Setting and Viewing Access ACLs

Objectives: Using the getfacl and setfacl commands on the command line to set and view ACLs on files.

Pre-Requisites: Reading and understanding all of the material and exercises presented in Section W23.9.3.1 above. Additionally, and perhaps most importantly, review the umask material we have presented in earlier chapters of the printed book, particularly the way umask is set depending upon whether or not you are working in a login or non-login terminal as posed in Chapter 2, Problem 25.. The premise of this example is that you are giving the following commands and working in a non-login terminal.

Background: The commands presented here allow an ordinary user to change the ACL’s of files and directories. For more information on the commands, see the man pages for **getfacl** and **setfacl** on your Linux system.

Requirements: Do the steps presented, in the order shown, to fulfill the requirements for this example.

1. Before creating a directory, check the default umask setting, then use the umask command to define which access permissions should be masked each time a file object is created. The command umask **007** sets the default permissions by giving the owner the full range of permissions (0), giving the group the full range of permissions (0), and giving other users no permissions at all (7). umask “masks” the permission bits, effectively turning them off. For a complete description of how this masking takes place, look at the umask man page on your system.

Then, create a directory and checking its traditional permissions with the ls -la command.

$ **umask**

0022

$ umask 007

$ mkdir acltest

$ ls -la acltest

total 8

drwxrwx--- 2 bob bob 4096 Oct 6 09:56 .

drwxr-xr-x 44 bob bob 4096 Oct 6 09:56 ..

$

2. These traditional permissions have an equivalent representation as ACL access permissions. Display the ACLs of the same directory using the getfacl command:

$ **getfacl acltest**

# file: acltest

# owner: bob

# group: bob

user::rwx

group::rwx

other::---

The first three output lines display the name, owner, and owning group of the directory. The next three lines contain the three ACL entries for owner, owning group, and other. This output is an example of a “minimum” ACL listing, as opposed to an “extended” ACL listing. The getfacl command produces the same information as ls -la command from step 1.

3. Give read, write, and execute access to user Mansoor in addition to the existing permissions. For that, the -m (modify) option of **setfacl** is used. Shown is the access ACL using the **getfacl** command.

$ **setfacl -m user:mansoor:rwx acltest**

$ **getfacl acltest**

# file: acltest

# owner: bob

# group: bob

user::rwx

user:mansoor:rwx

group::rwx

mask::rwx

other::---

Two additional entries have been added to the ACL: one is for user mansoor, and the the mask entry.

The “mask” entry is applied by the setfacl command and is governed by the group class permissions. It is composed of whatever is in the group class permission set, the logical “union” of all group class permissions.

4. Use the ls command to show changes to the traditional permissions of the directory.

$ **ls -la acltest**

total 8

drwxrwx---+ 2 bob bob 4096 Oct 6 09:56 .

drwxr-xr-x 44 bob bob 4096 Oct 6 09:56 ..

An additional ``+'' character is displayed after the traditional permissions of all files that have extended ACLs.

The permissions of the group class permissions include read and write access. Traditionally such file permission bits would indicate read and write access for the owning group. With ACLs, the effective permissions of the owning group are defined as the intersection of the permissions of the owning group and mask entries. The effective permissions of the owning group in the example are rwx, the same permissions as before creating additional ACL entries as done in step 3.

According to the output of the ls command in this step, the permissions for the mask entry include read and write access. Traditionally, such permissions would mean that the owning group also has write access to the directory. The access permissions for the owning group are the “union” of the owning group permissions and the mask permissions, which is rwx in our example .

5. The group class permissions are modified using the setfacl or chmod command. If we remove write access from the group class using the chmod command for example, we can use **ls -la** and **getfacl** to see how the traditional permissions and ACLs have been changed.

$ **chmod g-w acltest**

$ ls -la acltest

total 8

drwxr-x---+ 2 bob bob 4096 Oct 6 09:56 .

drwxr-xr-x 44 bob bob 4096 Oct 6 09:56 ..

$ getfacl --omit-header acltest

user::rwx

user:mansoor:rwx #effective:r-x

group::rwx #effective:r-x

mask::r-x

other::---

If an ACL entry contains permissions that are disabled by the mask entry, getfacl adds a comment that shows the effective set of permissions granted by that entry. If the owning group entry had write access, there would have been a similar comment for that entry.

After executing the chmod command to remove the write permission from the group class permissions, the output of the ls command shows us that the mask permissions have changed accordingly: write permission is now limited to the owner of the directory acltest. The output of the getfacl confirms this. This output includes a comment for all those entries in which the effective permission bits do not correspond to the original permissions, because they are reset according to the mask entry. The original permissions can be restored at any time with chmod g+w mydir, as is done in the next step.

6. Give write access to the group class again, and view the traditional and ACL access permissions.

$ **chmod g+w acltest**

$ ls -la acltest

total 8

drwxrwx---+ 2 bob bob 4096 Oct 6 09:56 .

drwxr-xr-x 44 bob bob 4096 Oct 6 09:56 ..

$ getfacl --omit-header acltest

user::rwx

user:mansoor:rwx

group::rwx

mask::rwx

other::---

In conclusion, an important thing to notice here is that after adding the write permission back to the group class, the access ACL defines the same permissions as before taking the permission away.

Example W26.22 Setting and Viewing Default ACLs

Objectives: To give some simple cases of setting and viewing default ACLs.

Pre-Requisites: Completion of Example W26.21. Additionally, and perhaps most importantly, review the umask material we have presented in earlier chapters of the printed book and as posed in Chapter 2, Problem 25., particularly the way umask is set depending upon whether or not you are working in a login or non-login terminal. The premise of this example is that you are giving the following commands and working in a non-login terminal.

Background: Directories can have a default ACL, which is a special kind of ACL defining the access permissions that objects in the directory inherit when they are created. A default ACL affects both sub-directories and files.

There are two ways in which the permissions of a directory's default ACL are passed to files and sub-directories: a sub-directory inherits the default ACL of the parent directory both as its default ACL and as its ACL, or a file inherits the default ACL as its ACL.

If the parent directory does not have a default ACL, the permissions defined by the umask are subtracted from the permissions that are given by the system call file access mode parameter. If a default ACL exists for the parent directory, the permission bits assigned to the new object correspond to the overlapping portion )or logical “intersection”) of the permissions of the system call-set mode parameter, discussed below, and those that are defined in the default ACL.

The previous example examined ACLs that define access permissions of file system objects. This example examines default ACL.

It is important to note that only directories can be associated with default ACLs. Regular files do not have default ACLs, because no file system objects can be created “inside” of regular files.

When a sub-directory is created, the new sub-directory inherits the parent directory's default ACL both as its access ACL and default ACL. Regular files inherit the default ACL of the directory they are created in as an access ACL.

System calls for accessing and changing file attributes include the following:

access(), chmod(), chown(), rename(), umask(), and utime().

The permissions of inherited access ACLs are modified by the file access mode parameter that each system call that creates file system objects exercises. The file access mode parameter established by these system calls contains the traditional nine permission bits that are the permissions of the owner, group, and other class permissions. The effective permissions of each class are set to the logical “intersection” of the permissions defined for any class in the ACL, and specified in the system call file access mode parameter.

The umask setting has no effect if a default ACL exists.

In-Chapter Exercises W26.21 and W26.22

Are there any system calls that directly affect ACLs, and what are their names and functions?

Give examples of system calls, that deal with the file access mode parameter.

Requirements: Do the steps presented, in the order shown, to fulfill the requirements for this example.

1. Create a group named development on your system, with the addgroup command.

$ **sudo addgroup development**

[sudo] password for bob: **QQQ**

Adding group `development' (GID 1002) ...

Done.

2. Add a default ACL to the directory acltest you created in Example W26.1, which gives read and execute privilege to members of the development group on the directory acltest. Finally, use the getfacl command to view the ACLs on that directory.

$ **setfacl -d -m group:development:r-x acltest**

$ **getfacl --omit-header acltest**

user::rwx

user:mansoor:rwx

group::rwx

mask::rwx

other::---

default:user::rwx

default:group::rwx

default:group:development:r-x

default:mask::rwx

default:other::---

Following the access ACL, the default ACL is printed with each entry prefixed with ``default:''. This output format is an extension to POSIX.1e that is typically available on Linux systems.

We have only specified an ACL entry for the development group in the setfacl command. The other entries required for a complete ACL have been copied from the access ACL to the default ACL.

The default ACL contains no entry for Mansoor, so Mansoor will not have access (except possibly through group membership or the other class permissions).

getfacl returns both the ACL and the default ACL. The default ACL is formed by all lines that start with default. Although we executed the setfacl command with an entry for the development group for the default ACL, setfacl automatically used all other entries from the ACL to create a default ACL. Default ACLs do not have an immediate effect on access permissions. They only are effective when directory file system objects are created. These new objects inherit permissions only from the default ACL of their parent directory.

3. This step will show that a subdirectory inherits ACLs as shown next. Unless otherwise specified, the mkdir command uses a value of 0777 as the mode parameter to the mkdir system call, which it uses for creating the new directory. Observe that both the access and the default ACL contain the same entries.

$ **mkdir acltest/acltest2**

$ **getfacl --omit-header acltest/acltest2**

user::rwx

group::rwx

group:development:r-x

mask::rwx

other::---

default:user::rwx

default:group::rwx

default:group:development:r-x

default:mask::rwx

default:other::---

As expected, the newly-created subdirectory named acltest2 has the permissions from the default ACL of the parent directory. The ACL of the sub-directory acltest2 is an exact reflection of the default ACL of its parent directory, acltest. And, most importantly, the default ACLs that this directory will pass to objects created in it is also the same.

4. In this step, we will create a file with the text editor vi, inside the directory acltest, and show how it inherits permissions.

$ **cd acltest**

~/acltest $ vi testfile1

In your favorite text editor, create and save a sample file named testfile1. We used vi.

~/acltest $

~/acltest $ **ls -la testfile1**

-rw-rw----+ 1 bob bob 21 Oct 6 13:07 testfile1

~/acltest $

~/acltest $ **getfacl --omit-header testfile1**

user::rw-

group::rwx #effective:rw-

group:development:r-x #effective:r--

mask::rw-

other::---

Conclusion:In this Example, we have given some simple cases of setting and viewing default ACLs, and have explained mask and effective ACLs.

In-Chapter Exercise W26.23

To get practical experience with setting and viewing ACLs, execute all of the steps of Examples W26.21 and W26.22, and verify that the output of each is the same on your Linux system.

W26.9.3.4 Setting NFSV4 ACLs on ZFS

It would be very useful to get some exposure to the NSFV4 ACL model, for three important reasons. First, that model allows an ordinary user to exercise a much wider range of permission controls than the default POSIX.1e ACL model on Linux. Second, in anticipation of the NFSV4 models’ adoption in Linux, you can experiment with the NFSV4 model, and gain some familiarity with the details of it. And third, since this section uses NSFV4 ACL commands to work with ZFS files in a Virtual Machine, you get the added advantages of exposure to ZFS and virtualization methodologies at the same time, here in this chapter.

It is highly recommended that you go ahead and learn the materials provided in the chapters on ZFS and Virtualization Methodologies at the Github site, before doing this section.

Example W26.24 Setting ACL Inheritance on ZFS Files

Objectives: To launch a TrueOS Virtual Machine using VirtualBox, and to explore setting characteristics of NSFV4 ACLs on that machine.

Pre-Requisites:

1. Completion of the chapters on ZFS and Virtualization Methodologies at the Github site,

2. Installation of VirtualBox on your Linux Mint system,

3. Download and installation of the most current TrueOS desktop operating system in VirtualBox,

running the TrueOS Virtual Machine instance

4. Creation of an ordinary user named mansoor using the adduser command in TrueOS

Background: This is necessarily a more complex and involved example, owing to the complexity of NFSV4 ACL’s in relation to the simpler POSIX.1e ACLs in the previous two examples. You can prepare for it by browsong through the ZFS shown in the chapter on ZFS at the Github site, and also looking more in-depth at how to install VirtualBox found in the chapter on virtualization at the Github site. But we only use a very basic number of ZFS commands, and expand upon the POSIX.1e model by showing more extensive setfacl and getfacl commands as they apply to NFSV4 ACLs. An excellent reference to use to understand the components of NFSV4 ACLs is the the Ubuntu man page at-

http://manpages.ubuntu.com/manpages/xenial/man5/nfs4\_acl.5.html

Furthermore, we provide the following tables of summary information on NFSV4 ACLs.

|  |  |
| --- | --- |
| ACL Entry Type | Description |
| owner@ | Specifies the access granted to the owner of the object. |
| group@ | Specifies the access granted to the owning group of the object. |
| everyone@ | Specifies the access granted to any user or group that does not match any other ACL entry. |
| user | With a user name, specifies the access granted to an additional user of the object. |
| group | With a group name, specifies the access granted to an additional group of the object. |

Table W26.10 NFSV4 ACL Entry Types

|  |  |  |
| --- | --- | --- |
| Access Privilege | Compact Access Privilege | Description |
| add\_file | w | Permission to add a new file to a directory. |
| add\_subdirectory | p | On a directory, permission to create a subdirectory. |
| append\_data | p | Permission to modify a file but only beginning from the end-of-file (EOF). |
| delete | d | Permission to delete a file. |
| delete\_child | D | Permission to delete a file or directory within a directory. |
| execute | x | Permission to execute a file or search the contents of a directory. |
| list\_directory | r | Permission to list the contents of a directory. |
| read\_acl | c | Permission to read the ACL (ls). |
| read\_attributes | a | Permission to read basic attributes (non-ACLs) of a file. |
| read\_data | r | Permission to read the contents of the file. |
| read\_xattr | R | Permission to read the extended attributes of a file. |
| synchronize | s | Permission to access a file locally at the server with synchronized read and write operations. |
| write\_xattr | W | Permission to create extended attributes or write to the extended attributes directory. |
| write\_data | w | Permission to modify or replace the contents of a file. |
| write\_attributes | A | Permission to change the times associated with a file or directory to an arbitrary value. |
| write\_acl | C | Permission to write the ACL or the ability to modify the ACL by using the chmod command. |
| write\_owner | o | Permission to change the file's owner or group. Or, the ability to execute the chown or chgrp commands on the file. |

Table W26.11 NFSV4 ACL Access Privileges

|  |  |  |  |
| --- | --- | --- | --- |
| Parent Directory Permissions | Target Object Permissions |  |  |
| " " (empty) | ACL allows delete | ACL denies delete | Delete permission unspecified |
| ACL allows delete\_child | Permit | Permit | Permit |
| ACL denies delete\_child | Permit | Deny | Deny |
| ACL allows only write and execute | Permit | Permit | Permit |
| ACL denies write and execute | Permit | Deny | Deny |

Table W26.12 NFSV4 ACL delete and delete\_child Permission Behaviors

|  |  |  |
| --- | --- | --- |
| Inheritance Flag | Compact Inheritance Flag | Description |
| file\_inherit | f | Only inherit the ACL from the parent directory to the directory's files. |
| dir\_inherit | d | Only inherit the ACL from the parent directory to the directory's subdirectories. |
| inherit\_only | I | Inherit the ACL from the parent directory. |
| no\_propagate | n | Only inherit the ACL from the parent directory to the first-level contents of the directory. |
| - | N/A | No permission granted. |
| successful\_access | S | Indicates whether an alarm or audit record should be initiated upon  a successful access. This flag is used with audit or alarm ACE  types. |
| failed\_access | F | Indicates whether an alarm or audit record should be initiated  when an access fails. This flag is used with audit or alarm ACE  types. |
| inherited | I | Indicates that an ACE was inherited. |

Table W26.13 NFSV4 ACL Inheritance Flags

In this example, you have the choice of either working directly on the command line in the TrueOS Virtual Machine instance, or of connecting to TrueOs running on your host system using ssh. The steps of the example can be done in either way; it is just a matter of your personal choice and what may be accessible to you on your host Linux Mint system.

Requirements: Do the following steps, in the order shown, to complete the requirements of this Example.

0. Install VirtualBox on your Linux Mint system, and then download the ISO file for the most current TrueOS version. Use the instructions available in the chapter on virtualization at the Github site to do this. After installing VirtualBox , install TrueOS as a VirtualBox guest system, also as shown in the chapter on virtualization at the Github site. Make sure that during the installation of the TrueOS guest, you enable sshd if you want to use ssh login to complete the steps shown below.

1. Once you have a working version of TrueOS, you can determine how ACLs are inherited on ZFS files and directories, in any particular filesystem, or “dataset”. The aclinherit property that we work with in the steps below is set to one of three possible values, using the zfs aclinherit command. These possible property value settings are “restricted”, “noallow”, and “discard”. The value “restricted” removes the write\_acl and write\_owner permissions when the ACL entry is inherited. The value “noallow”, applied to a filesystem, lets filesystem objects in it inherit only ACL entries that specify “deny” permissions. A value of “discard”, applied to a filesystem, means the filesystem objects in it cannot inherit any ACL entries.

The aclinherit property can be set globally on a filesystem, and in the steps below, we are working on the filesystem tank/usr/home/bob. Our default aclinherit mode was set at “noallow”. Therefore, we changed it to “restricted” to begin with.

Because the aclinherit property for any file system is set to some default mode, do the following to ascertain the aclinherit mode, and then set it to “restricted”.

You must change the data set name from tank/usr/home/bob to the whatever zfs data set you are working in on your system, and execute the **zfs set** command as superuser.

% **umask**

22

% **zfs get aclinherit tank/usr/home/bob**

NAME PROPERTY VALUE SOURCE

tank/usr/home/bob aclinherit noallow local

% **su**

Password: **yyy**

# **zfs set aclinherit=restricted tank/usr/home/bob**

# **exit**

exit

%

2. To grant ACLs that are inherited by files, do the following steps. They identify the file ACEs that are applied when the file\_inherit flag is set to allow.

In the following command, read\_data/write\_data permissions are added for files in the test2.dir directory for user bob so that he has read access on any newly created files.

% **mkdir test2.dir**

% **setfacl -m user:bob:read\_data/write\_data:file\_inherit:allow test2.dir**

% **getfacl test2.dir**

# file: test2.dir

# owner: bob

# group: bob

user:bob:rw------------:f------:allow

owner@:rwxp--aARWcCos:-------:allow

group@:r-x---a-R-c--s:-------:allow

everyone@:r-x---a-R-c--s:-------:allow

Notice how much more inclusive and detailed NFSV4 ACLs are, as shown by the output of the getfacl command. See the man page reference listed in Background for more information on NFSV4 ACLs, their format, and content, and also consult the four summary tables given in the Background material for this Example.

3. In this step, user bob's permissions are applied on a newly created file, named test2.dir/file.2 . The ACL inheritance granted, read\_data/write\_data:file\_inherit:allow from the setfacl command in step 2., means user bob can read and write to the contents of any newly created file in the directory test2.dir.

% **touch test2.dir/file.2**

**%** getfacl test2.dir/file.2

# file: test2.dir/file.2

# owner: bob

# group: bob

user:bob:rw------------:------I:allow

owner@:rw-p--aARWcCos:-------:allow

group@:r-----a-R-c--s:-------:allow

everyone@:r-----a-R-c--s:-------:allow

The inherit\_only permission, which is applied when the file\_inherit or dir\_inherit flags are set, is used to propagate the ACL through the directory structure.

4. For example:

% **mkdir test2.dir/subdir.2**

**% getfacl test2.dir/subdir.2**

# file: test2.dir/subdir.2

# owner: bob

# group: bob

user:bob:rw------------:f-i---I:allow

owner@:rwxp--aARWcCos:-------:allow

group@:r-x---a-R-c--s:-------:allow

everyone@:r-x---a-R-c--s:-------:allow

5. The following steps provide examples that identify the file and directory ACLs that are applied when both the file\_inherit and dir\_inherit flags are set.

In this step, user bob is granted read, write, and execute permissions that are inherited for newly created files and directories.

% **mkdir test3.dir**

% **setfacl -m user:bob:read\_data/write\_data/execute:file\_inherit/dir\_inherit:allow test3.dir**

% **getfacl test3.dir**

# file: test3.dir

# owner: bob

# group: bob

user:bob:rwx-----------:fd-----:allow

owner@:rwxp--aARWcCos:-------:allow

group@:r-x---a-R-c--s:-------:allow

everyone@:r-x---a-R-c--s:-------:allow

6. The I entry (standing for inherited) in the following output is informational, and indicates that the ACE is inherited.

% **touch test3.dir/file.3**

% **getfacl test3.dir/file.3**

# file: test3.dir/file.3

# owner: bob

# group: bob

user:bob:rwx-----------:------I:allow

owner@:rw-p--aARWcCos:-------:allow

group@:r-----a-R-c--s:-------:allow

everyone@:r-----a-R-c--s:-------:allow

7. In this step, user bob is granted read, write, and execute permissions that are inherited for newly created files, but are not propagated to subsequent contents of the directory, such as files added to the directory test4.dir.

% **mkdir test4.dir**

% **setfacl -m user:bob:read\_data/write\_data/execute:file\_inherit/no\_propagate:allow test4.dir**

% **getfacl test4.dir**

# file: test4.dir

# owner: bob

# group: bob

user:bob:rwx-----------:f--n---:allow

owner@:rwxp--aARWcCos:-------:allow

group@:r-x---a-R-c--s:-------:allow

everyone@:r-x---a-R-c--s:-------:allow

8. This step illustrates that bob's read\_data/write\_data/execute permissions are reduced in group@ and everyone@, or not propagated forward to a file, named file.4, created in the directory test4.dir.

% **touch test4.dir/file.4**

% **getfacl test4.dir/file.4**

# file: test4.dir/file.4

# owner: bob

# group: bob

user:bob:rwx-----------:------I:allow

owner@:rw-p--aARWcCos:-------:allow

group@:r-----a-R-c--s:-------:allow

everyone@:r-----a-R-c--s:-------:allow

9. The following steps describe changes in the aclinherit property values; the following command line examples illustrate ACL inheritance with the ACL inherit mode set to discard. If the ZFS aclinherit property on a file system is set to discard, then ACLs can potentially be discarded when the permission bits on a directory change.

One additional property of the dataset must also be checked at this point as well: the aclmode. This property, which may be set to either “discard”, “groupmask”, “passthrough”, or “restricted”, controls how an ACL is modified after a chmod command changes permissions on a file object. The default setting of aclmode is “discard”. A ZFS dataset with aclmode set to “discard” deletes all ACL entries that do not represent the appropriate mode of the file.

You must also, as above, change the data set name from tank/usr/home/bob to the whatever zfs data set you are working in on your system, and execute the zfs set command as superuser.

% **su**

Password: **yyy**

# **zfs set aclinherit=discard tank/usr/home/bob**

# **exit**

exit

% **mkdir test5.dir**

% **setfacl -m user:bob:read\_data/write\_data/execute:dir\_inherit:allow test5.dir**

% **getfacl test5.dir**

# file: test5.dir

# owner: bob

# group: bob

user:bob:rwx-----------:-d-----:allow

owner@:rwxp--aARWcCos:-------:allow

group@:r-x---a-R-c--s:-------:allow

everyone@:r-x---a-R-c--s:-------:allow

10. If, at a later time, you decide to tighten the permission bits on a directory, the non-trivial ACL is discarded. For example:

% **zfs get aclmode tank/usr/home/bob**

NAME PROPERTY VALUE SOURCE

tank/usr/home/bob aclmode discard default

% **chmod 744 test5.dir**

% **getfacl test5.dir**

# file: test5.dir

# owner: bob

# group: bob

owner@:rwxp--aARWcCos:-------:allow

group@:r-----a-R-c--s:-------:allow

everyone@:r-----a-R-c--s:-------:allow

In-Chapter Exercises W26.24 and W26.25

Why is the user class of ACL entries discarded in the above getfacl command?

Why are the group@ and everyone@ permissions reduced to r?

11. Finally, the following steps illustrate ACL inheritance with the ACL inherit mode set to “noallow”.

In it, two non-trivial ACLs with file inheritance are set. One ACL allows read\_data permission, and one ACL denies read\_data permission.

Again, you must change the data set name from tank/usr/home/bob to the whatever zfs data set you are working in on your system, and execute the zfs set command as superuser.

% **su**

Password: **yyy**

# **zfs set aclinherit=noallow tank/usr/home/bob**

# **exit**

exit

% **mkdir test6.dir**

% **setfacl -m user:bob:read\_data:file\_inherit:deny test6.dir**

% **setfacl -m user:mansoor:read\_data:file\_inherit:allow test6.dir**

% **getfacl test6.dir**

# file: test6.dir

# owner: bob

# group: bob

user:mansoor:r-------------:f------:allow

user:bob:r-------------:f------:deny

owner@:rwxp--aARWcCos:-------:allow

group@:r-x---a-R-c--s:-------:allow

everyone@:r-x---a-R-c--s:-------:allow

12. As the following step shows, when a new file, named “file.6” is created in the directory test6.dir, the ACL that allows read\_data permission to user mansoor is discarded.

% **touch test6.dir/file.6**

% **getfacl test6.dir/file.6**

# file: test6.dir/file.6

# owner: bob

# group: bob

user:bob:r-------------:------I:deny

owner@:rw-p--aARWcCos:-------:allow

group@:r-----a-R-c--s:-------:allow

everyone@:r-----a-R-c--s:-------:allow

In-Chapter Exercise W26.26

Execute all the steps shown in Example W26.24 on a VirtualBox guest TrueOs system, and verify that they give the same output.

W26.9.6.1 ufw and Netfilter Interface in Linux

Linux uses the ufw firewall to protect your system. By default, the firewall is configured to allow all outgoing connections, but to deny all incoming connection requests. The default configuration file for its rules is located in /etc/ufw/ufw.conf. If you want to know more about text-based modifications to firewall rules, see the **ufw** man page on your Linux system.

It is not absolutely necessary to change the firewall rules, and the basic details of using the command-line method of doing this are shown here.

Be very careful when adding custom rules or modifying the firewall, it may endanger your system’s security!

W26.9.6.2 Linux Uncomplicated Firewall (ufw)

The default firewall configuration utility for Linux Mint is the Uncomplicated Firewall, or ufw. Unlike the iptables firewall configuration utility, ufw is a much easier way to create an IPv4 or IPv6 host-based firewall. It is a rule-based system, which means that you create rules to control network connection access to your system. By default, ufw is disabled for example, on Linux Mint. But once enabled, it has a minimal set of rules, as part of its profile, that go into effect. ufw’s most basic application is to allow or deny access on ports or from specific IP addresses.

ufw is based upon the Netfilter interface to the Linux kernel, and particularly the filter table operations and protocols found in that interface. The rule format is also similar to the Packet Filter (PF) syntax in OpenBSD UNIX.

There is a GUI frontend for ufw, known as gufw. gufw can easily be installed on your system from the Software Manager. In the Questions and Problems section at the end of this chapter, we pose a scenario of firewall creation that uses gufw.

W26.9.6.2.1 Basic Syntax, Use Case and Rules Examples

An abbreviated listing of the ufw man page is as follows:

ufw

Purpose:

ufw is used to manage a Linux firewall, and provides an easy to use interface

for the creation of firewall rules. The rules use a basic syntax as shown below.

Syntax:

ufw [option[s]] [command] [rule[s]]

Output: New, modified, or deleted firewall rules to/from ports/ IP addresses, or devices.

Common Options:

--version show program's version number and exit

-h, --help show help message and exit

--dry-run don't modify anything, just show the changes

Common Commands:

enable reloads firewall and enables firewall on boot.

disable unloads firewall and disables firewall on boot

reload reloads firewall rules

default change the default connection policy

logging toggle and affect logging to journalctl

reset disables and resets firewall to defaults

status show status of firewall and rules

allow add a valid allow rule

deny add a valid deny rule

reject add a valid reject rule

limit add a valid limit rule

delete deletes a valid by specification

insert inserts a valid rule as a numbered rule

Common Rules:

ufw allow 22 allow tcp and udp port 22 to any IP address on a valid NIC

ufw deny proto tcp to any port 80 deny all connectons to tcp port 80

ufw allow in on enp2s0 to 192.168.0.6 proto tcp allow tcp connections through enp2s0 nic

to IP address 192.168.0.6

ufw limit ssh/tcp allow rate limiting on ssh to prevent brute-force attacks

ufw delete deny 80/tcp delete the rule denying tcp connections on port 80

ufw delete 3 delete rule 3, number determined with the status command

When you turn UFW on, it uses a default set of rules (profile) for the average home user. By default, all 'incoming' connections are denied. Following are a listing of the simple syntactic forms.

Use Case and Rules Examples-

To Check the Status, Enable, and Disable ufw-

Use Case 1.

To turn UFW on with the default set of rules:

$ **sudo ufw enable**

Use Case 2.

To check the status of UFW:

$ s**udo ufw status verbose**

[sudo] password: **QQQ**

Status: active

Logging: on (low)

Default: deny (incoming), allow (outgoing)

New profiles: skip

$

Note carefully that by default, deny is being applied to incoming connections.

Use Case 3.

$ **sudo ufw show raw**

You can also read the rules files in /etc/ufw (the files whose names end with .rules).

To Disable ufw

Use Case 4.

To disable ufw use:

$ **sudo ufw disable**

To Allow and Deny Connections by Creating Rules

The allow command rule syntax is as follows-

sudo ufw allow <port>/<optional: protocol>

Use Case 5.

To allow incoming tcp and udp packet on ephemeral port 32000

$ **sudo ufw allow 32000**

Use Case 6.

To allow incoming tcp packets on port 22

$ **sudo ufw allow 22/tcp**

Use Case 7.

To allow incoming udp packets on ephemeral port 16000

$ **sudo ufw allow 16000/udp**

The deny syntax is as follows-

sudo ufw deny <port>/<optional: protocol>

Use Case 8.

To deny tcp and udp packets on port 23

$ **sudo ufw deny 23**

Use Case 9.

To deny incoming tcp packets on port 23

$ **sudo ufw deny 23/tcp**

Use Case 10.

To deny incoming udp packets on port 23

$ **sudo ufw deny 23/udp**

Deleting an Existing Rule-

To delete a rule, prefix the original rule with delete. For example, if the original rule was:

ufw deny 8080/tcp, then to delete it, use:

Use Case 11.

$ **sudo ufw delete deny 8080/tcp**

To Allow by Service Name-

The allow syntax is as follows:

sudo ufw allow <service name>

Use Case 12.

To allow ssh by name

$ **sudo ufw allow ssh**

[sudo] password for bob: **QQQ**

Rule added

Rule added (v6)

Check the status of rules after the above command-

$ sudo ufw status

Status: active

To Action From

-- ------ ----

22 ALLOW Anywhere

22 (v6) ALLOW Anywhere (v6)

Deny by Service Name-

The deny syntax is as follows:

sudo ufw deny <service name>

Use Case 13.

To deny ssh by name

$ **sudo ufw deny ssh**

ufw Logging

System logs can record events that impact ufw, as well as the effects network connections and traffic have on other components of the system.

Use Case 14.

To enable logging use:

$ **sudo ufw logging on**

Use Case 15.

To disable logging use:

$ sudo ufw logging off

In-Chapter Exercise W26.27

What is the biggest advantage of logging ufw events?

W26.9.6.2.2 Advanced ufw Syntax

The advanced syntax, or long form syntax, can contain more command arguments. These can specify objects like source and destination addresses, ports and protocols.

Allow Access

This section shows how to allow specific access.

Allow by Specific IP

The allow syntax is as follows:

sudo ufw allow from <ip address>

Use Case 14.

To allow packets from 192.168.0.8:

$ **sudo ufw allow from 192.168.0.8**

To Add a Rule to Allow by Subnet

Use Case 15.

You may use a subnet mask as follows:

$ **sudo ufw allow from 192.168.1.0/24**

To Add a Rule to Allow by specific port and IP address

The general syntax for adding an allow rule is as follows:

sudo ufw allow from <target> to <destination> port <port number>

Use Case 16.

Allow IP address 192.168.0.4 access to port 22 for all protocols

$ **sudo ufw allow from 192.168.0.4 to any port 22**

Use Case 17.

To add a range of ports into a rule, then the following command example will do that:

$ **sudo ufw allow 32000:32500/tcp**

To allow to a specific port, IP address, and using a specific protocol

The general syntax for specific ports, IP addresses, and protocols is:

sudo ufw allow from <target> to <destination> port <port number> proto <protocol name>

Use Case 18.

To allow IP address 192.168.0.4 access to port 22 using TCP

$ **sudo ufw allow from 192.168.0.4 to any port 22 proto tcp**

Deny Access

To deny connections by designating a specific IP address

The general format for denying to a specific IP address is:

sudo ufw deny from <ip address>

Use Case W26.

To block connections from 192.168.0.12:

$ **sudo ufw deny from 192.168.0.12**

To deny by a specific port and IP address

The general syntax for denying is:

sudo ufw deny from <ip address> to <protocol> port <port number>

Use Case W26.

To deny a connection from IP address 192.168.0.7 to port 22 for all protocols

$ **sudo ufw deny from 192.168.0.7 to any port 22**

Specification by Numbered Rules

Listing rules with a reference number

Use Case 21.

Use the status numbered rule construct to show the order and id number of existing rules:

$ **sudo ufw status numbered**

Status: active

To Action From

-- ------ ----

[ 1] 65000 DENY IN 16000

[ 2] 65000 (v6) DENY IN 16000 (v6)

$

To delete a numbered rule

Use Case 22.

After executing the command in Use Case 21., you can delete rules using the numbers found at the left margin of the output. The following will delete the first rule, and the remaining rules will shift up in numbering to fill in the list.

$ **sudo ufw delete 1**

W26.9.6.2.3 An Extended Example of Applying ufw Rules

The following example applies some of the previous Use Case commands in an extended and more complex arrangement. Basically, it applies some user-defined rules in the first two steps, and then shows a means of changing those rules. For a further description of rule syntax and interaction, we refer you on to the existing documentation for the Netfilter interface.

Example W26.25

1. Deny connection from the IP address 192.168.0.8 for ssh-

$ **sudo ufw deny from 192.168.0.8 to any port 22**

Rule added

$

2. Allow ssh connections from all other IP addresses on our LAN, using the tcp protocol-

$ **sudo ufw allow from 192.168.0.0/24 to any port 22 proto tcp**

Rule added

3. Check the status of ufw rules-

$ **sudo ufw status**

Status: active

To Action From

-- ------ ----

22 DENY 192.168.0.8

22/tcp ALLOW 192.168.0.0/24

$

4. To test the above rules, first login from the machine with the IP address of 192.168.0.8 on your LAN-

$ **ssh bob@192.168.0.6**

ssh: connect to host 192.168.0.6 port 22: Connection timed out

$

To further test, you can login from a machine on your LAN that is not 192.168.0.8. You should be able to ssh login from any other machine on your LAN within the range specified.

5. Add an additional deny rule to deny an ssh connection from 192.168.0.31. Since the rules are evaluated in order, from 1 to n, an addition of a deny rule at this point would make it rule #3. Therefore, according to the precedents of chain and rule evaluation designated in the Netfilter protocols, it is not evaluated. The following steps are one way of moving a rule into the chain of evaluation.

So first delete rule #2, the allow rule for all IP addresses on the LAN-

$ **sudo ufw delete allow from 192.168.0.0/24 to any port 22 proto tcp**

Rule deleted

$

Check the status of rules-

$ **sudo ufw status**

Status: active

To Action From

-- ------ ----

22 DENY 192.168.0.8

$

6. Add in the rule denying ssh connections from 192.168.0.31, using tcp protocol-

$ **sudo ufw deny from 192.168.0.31 to any port 22**

Rule added

$

7. Add back the rule allowing ssh from all other IP addresses on the LAN, at the end of the chain-

$ **sudo ufw allow from 192.168.0.0/24 to any port 22 proto tcp**

Rule added

$

8. Check the status of rules-

$ **sudo ufw status**

Status: active

To Action From

-- ------ ----

22 DENY 192.168.0.8

22 DENY 192.168.0.31

22/tcp ALLOW 192.168.0.0/24

$

9. Test from 192.168.0.8 and 31, and from some other IP on the LAN to verify the effectiveness of the rules.

In-Chapter Exercises W26.28 and W26.29

W26.28. What ufw rule could have been used at step 5. that would have the same effect as steps 6 through 8?

W26.29. Give all the ufw commands that would erase all the rules established here.

W26.9.6.2.4 Interpreting ufw Log Entries in the systemd Journal

Question: Why would you want to look at log entries generated by ufw?

Answer: Someone is trying to crack into your machine from your LAN or the Internet, and you want to have information about their method of attack.

This section details an example procedure for generating ufw-blocked messages in the log kept by the systemd Journal. We then give brief descriptions of the details of the components of the log entry. The Journal, and the journalctl command in particular, allows you to view consolidated log messages from a variety of sources on your system, all using one command. We give details of the systemd Journal, and the journalctl command, in Section W26.8, and more fully in the chapter on systemd at the Github site.

Example W26.26 ufw Log Entries

1. You have to set up a deny rule that blocks connections. In this example, to set up a rule that blocks connections from 192.168.0.8 on our LAN, we give the following ufw rule:

$ **sudo ufw deny in log-all from 192.168.0.8**

Rule added

$

The log-all parameter turns on full ufw logging. Since this parameter generates quite a bit of log entry data, it would be advisable to turn logging off promptly as shown in step 5.

2. When we view ufw’s verbose status, it gave us this output:

$ **sudo ufw status verbose**

[sudo] password for bob: **QQQ**

Status: active

Logging: off

Default: deny (incoming), allow (outgoing), disabled (routed)

New profiles: skip

To Action From

-- ------ ----

Anywhere DENY IN 192.168.0.8 (log-all)

3. Attempt to login to the current machine that you set up the rule on in step 1., from some other place on your LAN or possibly from the Internet. We attempted an ssh login from another machine on our LAN with an IP address of 192.168.0.8. Of course, this login attempt assumes that sshd is running on the system you created the ufw rule on in step 1. In the Questions and Problems section at the end of this chapter, we pose a problem where you use different forms of remote login, and note the differences in log entries for those types of attempted connection.

4. Immediately examine the Journal log, using the journalctl – command. The –f option of the journalctl command shows you log entries generated in real time. This yields log entries similar to the one in the next step. This is the contemporary method of examining system logging, the mechanics of which are shown in detail in the chapter at the Github site on systemd.

5. Turn ufw logging off with the following command:

$ **sudo ufw logging off**

Logging disabled

$

6. Here is a sample of one of the log entries we got by doing the . Following it, we will comment on some of the items in it that would be useful for a system administrator to use when analyzing the entry.

Log Entry

Nov 01 11:23:33 bob-ProLiant-MicroServer kernel: [UFW BLOCK] IN=enp2s0 OUT= MAC=ff:ff:ff:ff:ff:ff:b8:ac:6f:9a:80:dc:08:00 SRC=192.168.0.8 DST=192.168.0.6 LEN=194 TOS=0x00 PREC=0x00 TTL=64 ID=53133 DF PROTO=TCP SPT=17500 DPT=17500 LEN=174

An explanation of the components of this entry is as follows:

Date - Nov 01 11:23:33

Places entries in time order, so you can see the progress of what connections have been done, or most importantly, what connections have been attempted. Since this log entry in the Journal was generated and viewed in real time using the journalctl –f command, it is a very recent excerpt from the real time display of the Journal log.

Hostname - bob-ProLiant-MicroServer kernel

The server’s hostname, useful to identify the machine you are reading the Journal on, if you are dealing with several remote machines and are monitoring their Journal entries.

Event - [UFW BLOCK]

A short description of the logged event; depending on the level of error generated, this could be just an Audit.

IN - enp2s0

Incoming connection at the NIC enp2s0. Useful if the machine you are monitoring the log entry on has several NIC’s, connected to different branches on your network.

OUT- blank

IN our log entry, blank because it is not an outgoing connection attempt.

MAC- ff:ff:ff:ff:ff:ff:b8:ac:6f:9a:80:dc:08:00

A 14-byte combination of the Destination MAC, Source MAC, and [EtherType](https://help.ubuntu.com/community/EtherType) fields, following Ethernet protocol.

SRC - 192.168.0.8

The source IP, in our case the IP address of the machine on the LAN we attempted to connect from. If this source is an IPV4 address, it can tell you how to apply an additional set of filtering rules to prevent unwanted connections.

DST - 192.168.0.6

This indicates the destination IP. In this case, the IP address of the system we created the

deny rule on.

LEN - 194

This indicates the length in bytes of the packet. Can tell you about what type of entry connection has been attempted.

The next five entry components are not important in basic log interpretation:

TOS - 0x00

PREC -0x00

TTL - 64

ID – 53133

DF

PROTO - TCP

The protocol of the packet - TCP or UDP. TCP in our case, because it was an ssh connection.

SPT - 17500

The port the source sent the packet through from. Interesting here that the ssh client on 192.168.0.8 used an ephemeral port to attempt the connection on.

DPT - 17500

The port on the destination the packet was sent to. Again, interesting that the packet was routed to this ephemeral port on the destination system, and ufw blocked it.

LEN - 174

Not important in basic log interpretation.

W26.9.7.2 Whole Disk Encryption at Installation

It is possible at the time you install Linux Mint to specify that you want the system disk encrypted, so that no one can gain access to it without submitting a “passphrase”, or cryptographic security key. This way, if someone boots the hardware from another medium, such as a USB-attached thumb drive or a DVD, they cannot get access to your system disk or anything on it without also submitting the correct passphrase. It is also possible to encrypt an entire disk after installation as well.

In the following example, we show the procedures for encrypting the entire boot disk at the time you install the system onto it.

Example W26.11 Linux Mint Encrypted Disk Installation

Objectives: To show how to encrypt the entire system disk during the installation of Linux Mint.

Pre-Requisites: Having an ISO-image on DVD of the Linux Mint you want to install on your hardware.

Requirements: Do the steps below in the order shown to complete the requirements for this example.

1. Boot your system to the LiveDVD. Once Linux Mint is running, double-click on the Install Linux Mint icon on the desktop. The first Installer menu appears.

2. Choose Language, and optionally, third-party software on their Installer menu screens.

3. On the Installation type screen, put a check mark in the box next to Encrypt the new Linux Mint installation for security. Notice that the Erase Disk and Use LVM boxes will be checked as well. Then click on the Install Now button.

4. The Choose a security key screen appears. Type in a key, and confirm it. Make sure you write down the security key you have used! Click on the Install Now button.

5. In the Write the changes to disk screen, make the Continue choice.

6. Proceed through the rest of the Installation menus, and finally reboot.

7. In the rebooting process, you will be asked to decrypt the main partition of your disk drive before you can continue with the boot. Type in the security key you designated in step 4.. The system will then boot.

W26.9.7.3 Encrypting a Directory Using tar and gpg

The objective in this section is to illustrate not only encryption and compression of an entire directory, but also to further show use of the tar command to preserve the directory structure.

Example W26.12 Directory Encryption

Objectives: To take a directory, and treat it with tar and gpg, to both compress and encrypt it.

Pre-Requisites: Having a directory, containing files you want to protect with encryption using gpg.

Requirements: Do the following steps in the order presented to fulfill the requirements of this example.

1. While in your home directory, use the following command to tar/compress and encrypt a directory of your choice. In our case the directory was named 18\_C:

$ **tar --gzip --create --verbose --file - 18\_C | gpg -c > 18\_C.tar.gz.gpg**

18\_C/

18\_C/cpy.c

18\_C/foobar

Output Truncated…

gpg: directory `/home/bob/.gnupg' created

gpg: new configuration file `/home/bob/.gnupg/gpg.conf' created

gpg: WARNING: options in `/home/bob/.gnupg/gpg.conf' are not yet active during this run

gpg: keyring `/home/bob/.gnupg/pubring.gpg' created

Enter passphrase: **xxxx**

Repeat passphrase: **xxxx**

$

Remember to write down the passphrase so you can decrypt the directory! The tarred, compressed, and encrypted directory is named 18\_C.tar.gz.gpg.

2. This step decrypts and “untars” and uncompresses the encrypted directory and all its files.

$ **gpg --decrypt 18\_C.tar.gz.gpg | tar --extract --gunzip --verbose --file -**

gpg: keyring `/home/bob/.gnupg/secring.gpg' created

gpg: AES encrypted data

gpg: gpg-agent is not available in this session

gpg: encrypted with 1 passphrase

Enter passphrase: **xxxx**

18\_C/

18\_C/cpy.c

18\_C/foobar

Output truncated…

$

In-Chapter Exercise w26.30

Use the methods of Example W26.12 to encrypt, but not compress, an important directory under your home directory on your system. Then, decrypt that directory using gpg. Where are the decrypted files placed, for example, is the original directory you encrypt overwritten?

W26.9.7.4 Encrypting a User File

There are times when an ordinary user might want to keep one or more files secure by encrypting them. This might be true when you want to archive the files using a USB thumb drive or other removable medium, and want to avoid loss of privacy if you lose the thumb drive. The following example illustrates the use of gpg to encrypt a single important file.

Example W26.13 Encryption of a Single File

Objective: To encrypt an important file in your home directory.

Pre-Requisites: Doing Example W26.12, having a file in your home directory you want to encrypt.

Requirements: Do the following steps in the order presented to fulfill the requirements of this example.

1. Create a file with vi, that contains the string “This is a test file for encryption.”

$ **vi newfile**

Or use some existing file.

2. Use gpg to encrypt the file. Make sure to enter a passphrase you can remember, or even write down the passhrase!

$ **gpg --symmetric --cipher-algo AES256 newfile**

gpg: directory `/home/bob/.gnupg' created

gpg: new configuration file `/home/bob/.gnupg/gpg.conf' created

gpg: WARNING: options in `/home/bob/.gnupg/gpg.conf' are not yet active during this run

gpg: keyring `/home/bob/.gnupg/pubring.gpg' created

Enter passphrase: **yyyy**

Repeat passphrase: **yyyy**

$

You can replace the AES56 with many other alogrithms, such as IDEA, 4DES, CAST5, BLOWFISH, AES, etc..

3. Examine the contents of newfile. It’s unreadable.

$ **more newfile.gpg**

# ##h�D=��(�`�\#�

33���vެ#K�Y+���m#�v�(7��-#�Ymښ�'&g@\�#�4�8H׺����q#G1{zG#��H�#�$+��<�j# l�d

/#�w�OA

4. Delete the original, unencrypted file.

$ **rm newfile**

5. Decrypt the file we encrypted in step 2..

$ **gpg newfile.gpg**

gpg: AES256 encrypted data

gpg: gpg-agent is not available in this session

gpg: encrypted with 1 passphrase

Enter passphrase: **yyyy**

6. Examine what is in your current working directory.

$ **ls**

Desktop Downloads newfile Pictures Templates

Documents Music newfile.gpg Public Videos

Notice both the newly decrypted file and the encrypted file are both there.

7. Examine the contents of the newly decrypted file.

$ **more newfile**

This is a test file for encryption.

$

Notice its contents are the same as the original.

In-Chapter Exercise W26.31

Use the methods of Example W26.13 to encrypt an important file in your home directory on your system. Then, decrypt that file using gpg. Where is the decrypted file placed, for example, is the original file you encrypted overwritten?

W26.9.7.5 Encrypting a USB Thumb Drive

Since USB removable media are easily subject to loss of privacy issues, we present this method of encrypting an entire USB thumb drive. An important aspect of this method is that it may be applied to other partitions, which may be on other persistent media attached to your system.

Example W26.14 How to Encrypt a USB Thumb Drive

Objectives: To encrypt an entire single-partition USB thumb drive.

Pre-Requisites: Having a blank, automatically-mounted USB thumb drive for use on your system.

Background: The facilities that allow this procedure to be done are the cryptsetup utility, which is available by default on Linux Mint, and the Disks utility. A properly single-partitioned and formatted USB thumb drive, which is usable on your Linux Mint system, automounts, and an icon for it appears on the desktop when it is inserted into a USB port. This example allows you to encrypt that thumb drive, so that in case it is lost, the private data you subsequently put on it is not compromised. Also, if you insert it in another computer, you can type-in the encryption passphrase, and it will be unlocked and usable on that second computer as well. This feature allows you to safely transport the thumb drive between computer sites.

Requirements: Do the following steps, in the order presented, to fulfill the requirements of this example.

1. Insert your USB thumb drive into a USB port, and then Launch the Disks utility from the Mint Menu>Accessories>Disks. In the left side of the Disk utility window, your disk drives will be shown.

2. Click on the USB thumb drive in the left side window. On the right side of the window, information about your USB thumb drive will be shown. In the little black square under the block showing the thumb drive partition, make the choice “unmount selected partition”.

3. Click on the black gear box square under the block showing the thumb drive partition, and make the choice Format Partition. In the Format Volume window that opens on screen,

a. pull down the menu Type, and choose Encrypted, compatible with Linux systems (LUKS + Ext4).

b. Give the volume a name of your choice.

c. Type in a Passphrase, and Confirm Passphrase. Make sure you can remember the passphrase!

d. Click on the Format… button in the lower-right of the Format Volume window. When the “Are you sure you want to format the volume?” warning window appears, click on the Format button.

e. Wait until the filesystem is built, then quit the Disk utility using the icon in the upper-left corner of the Disk utility window.

4. Withdraw the USB thumb drive from the computer, and reinsert it. A window appears on screen, asking you to “Enter a passphrase to unlock the volume”. In the password field, enter the passphrase you typed-in in step 3.c.. If you always want the passphrase to be remembered whenever you plug this USB thumb drive into the computer, check off the Remember forever choice. Then click on the connect button. The USB thumb drive will be mounted, and a Nemo file window will open into it.

5. You are now free to add directories and files to the thumb drive. Notice that on the Linux Mint desktop, the icon for the USB thumb drive appears with an unlocked u-lock figure shown on it.

6. To eject the USB thumb drive, right-click on it, and make the eject choice.

7. When mounting this USB thumb drive on another computer, you will be asked the passphrase before it can be unlocked on the other computer.

In-Chapter Exercise W26.32

Use the methods of Example W26.14 to encrypt a USB thumb drive you attach to your system. Then, detach the thumb drive, and mount it on another Linux Mint computer, and use the passphrase to unlock it on that computer.

W26.9.8.2 How a Process Gets Its Credentials

As detailed in Chapters 10 and 16, each Linux process, or task, has a number of data structure identifiers that are created by the relevant system calls, to maintain that process’s autonomy and the steady-state of the multi-programming model. These identifiers are critical to Linux authentication checks on processes. As examples, a child process created by the fork system call inherits its parent's session ID and process group ID. A process's session and group ID are maintained across an execve system call. These identifiers include objects such as Process ID (PID), Parent Process ID (PPID), Process Group ID and Session ID, and User and Group Identifiers. All of these identifiers, and the data structures associated with them, play directly into the Linux security model. Most important to our notion of security are User and Group Identifiers .

In terms of process authentication checks, User and Group Identifiers are listed as real user and group ID, effective user and group ID, the saved set-user-ID and set-group-id, the Linux-specific filesystem user and group ID, and supplementary group IDs.

The interaction of assignments of these identifiers is complex, but can be prioritized as follows:

\* Privileged processes (with UID=0) have access to everything.

\* The effective UID is the same as the owner’s UID, access is given by the owner’s permissions on a file object.

\* The effective GID, or supplementary process GID’s match the owning group GID, permission is given by the owning group permissions on the file object.

\* Finally, access permissions are given by the “other” permission on the file object.

Additionally, a process's user IDs are applied in other ways, such as:

When determining the permissions for sending signals, such as the kill signal.

When determining the permissions for setting process-scheduling, such as setting the nice value.

When checking resource limits, and the number of inotify instances possible for the process.

For more information on credentials, see the man page for credentials on your Linux system.

W26.9.8.3 Process Capabilities

There is also a way to assign credentials to processes, through a scheme known as “capabilities”. Capabilities use a finer-grained technique to limit or grant credentials to processes you want to give privileges to. This is in contrast to the traditional RBAC method of using sudo to give coarse-grained privileges to a process, or set of processes initiated by an unprivileged user.

It is possible to use Bash to assign and view process capabilities, using the setcap and getcap commands. Essentially, a program and its processes have individualized and custom high-level privileges, rather than a blanket or generalized high-level privilege, such as a user ID of 0 (root). We show an example of this technique in Example W26.27.

You can also assign capabilities from within system programs, using the appropriate system calls from the libcap API.

For each process, the kernel maintains three capability sets which contain any of the capabilities specified in Table W26.14, if they are are enabled. At the same time, if a file has associated capability sets, then these sets are used to determine the capabilities that are given to a process if it uses the exec() system call on that file. The three sets for process and file, arranged in order from least to most inclusive, are:

Inheritable: These are capabilities that can be passed on to the permitted set when a process uses the exec() system call when accessing a file.

Effective: These are the capabilities used to perform privilege checking for the process.

Permitted: These are the maximum capabilities, formed from effective and inherited capabilities, that a process can actually use.

The possible file capability set and its contents gives an executable file process capabilities.

It assigns a group of capabilities that are given to the process’s permitted capability set during a system call to exec() made by the process.

Think of the process and file sets as the raised ridges on a house key (the process set), and as tumblers in the lock on the front door (the file set). If the key ridges match the tumbler settings, you are granted access and can open the door lock with the key. The process can be authenticated via its capability set checked against the capability set on any file it wants to use exec() on .

The simplified syntax of the setcap command is as follows:

setcap [-options] (capabilities) filename

The simplified syntax of the getcap command is as follows:

getcap [-options] filename

See the man pages on your system for more details of the setcap and getcap commands.

Example W26.27 Setting Capabilities on the Command Line

Objectives: To set and view a process capability using the setcap and getcap commands.

Pre-Requisites: Having a root-privileged account, and also having created a standard, non-privileged account (which we named lowly, with a password low), that is not a member of the root, or sudoers group. In the steps below, when you are required to use the privileged account, that account should be a part of the sudoers group to execute the sudo command as shown.

Background: We modify a copy of the Bash external command /bin/ping program, so that when it is used from an unprivileged account, it can execute without setting the SUID permission bit.

To quickly and easily switch between privileged and unprivileged accounts, make use of the virtual terminals available in Linux with the **<Ctrl>+<Alt>+Function key(s)**. On our Linux Mint system, **<Ctrl>+<Alt>+F1** enabled us to log into the unprivileged account in a virtual, text-only terminal, and the default tty into the privileged account was available at **<Ctrl>+<Alt>+F7.**

Requirements: Do the following steps in the order presented to meet the requirements of this example.

1. Use **<Ctrl>+<Alt>+F1** to login to the unprivileged account, and use the **ls –la** command to see the permission bits set on the system-wide /bin/ping executable program. Then test ping, by executing the ping command to connect to localhost:

$ **ls -la /bin/ping**

-rwsr-xr-x 1 root root 44168 May 7 2014 /bin/ping

$ /bin/ping localhost

PING localhost (127.0.0.1) 56(84) bytes of data.

64 bytes from localhost (127.0.0.1): icmp\_seq=1 ttl=64 time=0.048 ms

64 bytes from localhost (127.0.0.1): icmp\_seq=2 ttl=64 time=0.053 ms

64 bytes from localhost (127.0.0.1): icmp\_seq=3 ttl=64 time=0.053 ms

<Ctrl>+C

--- localhost ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 1998ms

rtt min/avg/max/mdev = 0.048/0.051/0.053/0.006 ms

The “s” bit of the file’s permission mode is the setuid-executable bit, and as an unprivileged user, you are able to use ping because of this.

2. While still in the unprivileged account, copy the /bin/ping file to the unprivileged user home directory. As the command below shows, ping loses its privilege and no longer works, as shown by the following commands:

$ **cp /bin/ping .**

$ **ls -la ping**

-rwxr-xr-x 1 lowly lowly 44168 Dec 25 15:16 ping

$ **./ping localhost**

ping: icmp open socket: Operation not permitted

$

ping requires privilege to write the network packets that are used to probe the network. The Linux kernel checks to see whether this copy of ping in the unprivileged account is capable(CAP\_NET\_RAW), which means cap\_effective (pE) for the current process includes CAP\_NET\_RAW. By default, root gets all effective capabilities, so it defaults to having more-than-

enough privilege to successfully use ping. Similarly, when setuid-root, the /bin/ping version is also

overly privileged. If some attacker were to discover a new buffer-overflow or more subtle bug in the ping application, then they might be able to exploit it to invoke a shell with root privilege.

Filesystem capability support adds the ability to bestow just-enough privilege to the ping application.

3. To give the unprivileged ping just enough privilege to execute the ping command in it’s home directory, switch to the privileged account by using the key combination **<Ctrl>+<Alt>+F7**. Use the utilities from libcap to do the following:

$ **sudo setcap cap\_net\_raw=ep /home/lowly/ping**

$ **sudo getcap /home/lowly/ping**

ping = cap\_net\_raw+ep

$

What the first command does is add a limited permitted (the p argument) capability for CAP\_NET\_RAW, and also sets the effective bit (e), to automatically raise this effective bit in the unprivileged ping process at the time the ping command is executed.

4. Switch back to the unprivileged account using <Ctrl>+<Alt>+F1, and then use the following command to test the local ping command:

$ **./ping localhost**

PING localhost (127.0.0.1) 56(84) bytes of data.

64 bytes from localhost (127.0.0.1): icmp\_seq=1 ttl=64 time=0.049 ms

64 bytes from localhost (127.0.0.1): icmp\_seq=2 ttl=64 time=0.053 ms

64 bytes from localhost (127.0.0.1): icmp\_seq=3 ttl=64 time=0.053 ms

64 bytes from localhost (127.0.0.1): icmp\_seq=4 ttl=64 time=0.058 ms

Press **<Ctrl>+C**

--- localhost ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 2997ms

rtt min/avg/max/mdev = 0.049/0.053/0.058/0.006 ms

Unlike setting the setuid-root permission bit , the ping in the unprivileged account after step 3. is not given blanket privileges, such as any privilege to modify a file that is not owned by the calling user, or to deploy a kernel module in some way. There is no direct way to subvert this localized ping program.

Conclusion: Instead of giving a process blanket root privileges, Linux capabilities allow you to only give an application and its processes just enough privilege to accomplish its objectives.

In-Chapter Exercise W26.33

Following the general procedure of Example W26.27, give an unprivileged user the capability of changing both owner and group of any file on your system. Utilize the information found in Table W26.14.

|  |  |
| --- | --- |
| Capability | What It Applies To |
| CAP\_AUDIT\_CONTROL | Enable and disable kernel auditing |
| CAP\_AUDIT\_READ | Allow reading the audit log via a multicast netlink socket. |
| CAP\_AUDIT\_WRITE | Write records to kernel auditing log. |
| CAP\_BLOCK\_SUSPEND | Employ features that can block system suspend. |
| CAP\_CHOWN | Make arbitrary changes to file UIDs and GIDs (see chown(2)). |
| CAP\_DAC\_OVERRIDE | Bypass file read, write, and execute permission checks. |
| CAP\_DAC\_READ\_SEARCH | Bypass file read permission checks and directory read and execute permission checks; |
| CAP\_FOWNER | Bypass permission checks on operations that normally require the filesystem UID of the process to match the UID of the file. |
| CAP\_FSETID | Don't clear set-user-ID and set-group-ID mode bits when a file is modified; set the set-group-ID bit for a file whose GID does not match the filesystem or any of the supplementary GIDs of the calling process. |
| CAP\_IPC\_LOCK | Lock memory (mlock(2), mlockall(2), mmap(2), shmctl(2)). |
| CAP\_IPC\_OWNER | Bypass permission checks for operations on System V IPC objects. |
| CAP\_KILL | Bypass permission checks for sending signals (see kill(2)). |
| CAP\_LEASE | Establish leases on arbitrary files (see fcntl(2)). |
| CAP\_LINUX\_IMMUTABLE | Set the FS\_APPEND\_FL and FS\_IMMUTABLE\_FL inode flags . |
| CAP\_MAC\_ADMIN | Override Mandatory Access Control (MAC). |
| CAP\_MAC\_OVERRIDE | Allow MAC configuration or state changes. |
| CAP\_MKNOD | Create special files using mknod(2). |
| CAP\_NET\_ADMIN | Perform various network-related operations. |
| CAP\_NET\_BIND\_SERVICE | Bind a socket to Internet domain privileged ports (port numbers less than 1024). |
| CAP\_NET\_BROADCAST | (Unused) Make socket broadcasts, and listen to multicasts. |
| CAP\_NET\_RAW | Use RAW and PACKET sockets. |
| CAP\_SETGID | Make arbitrary manipulations of process GIDs and supplementary GID list. |
| CAP\_SETFCAP | Set file capabilities. |
| CAP\_SETPCAP | If file capabilities are not supported: grant or remove any capability in the caller's permitted capability set to or from any other process. |
| CAP\_SETUID | Make arbitrary manipulations of process UIDs. |
| CAP\_SYS\_ADMIN | Perform a range of system administration operations. |
| CAP\_SYS\_BOOT | Use reboot(2) and kexec\_load(2). |
| CAP\_SYS\_CHROOT | Use chroot(2). |
| CAP\_SYS\_MODULE | Load and unload kernel modules. |
| CAP\_SYS\_NICE | Raise process nice value (nice(2), setpriority(2)) and change the nice value for arbitrary processes. |
| CAP\_SYS\_PACCT | Use acct(2). |
| CAP\_SYS\_PTRACE | Trace arbitrary processes using ptrace(2). |
| CAP\_SYS\_RAWIO | Perform I/O port operations (iopl(2) and ioperm(2)). |
| CAP\_SYS\_RESOURCE | Use reserved space on ext2 filesystems, and other resource limits. |
| CAP\_SYS\_TIME | Set system clock. |
| CAP\_SYS\_TTY\_CONFIG | Use vhangup(2); employ various privileged ioctl(2) operations on virtual terminals. |
| CAP\_SYSLOG | Perform privileged syslog(2) operations. |
| CAP\_WAKE\_ALARM | Trigger something that will wake up the system. |

Table W26.14 Linux Capabilities

W26.9.9 Namespaces and User Namespaces

These are security-related features built into the Linux kernel that differentiate it from UNIX kernels, particularly FreeBSD and Solaris. They are methods of creating a virtual environment for a group of processes. The kinds of resources that can be virtualized in these environments for groups of processes are currently process IDs, hostnames, user IDs, network access, interprocess communication, and filesystems. Namespaces are a fundamental component of the container virtualization technique we show in Chapter W23 Virtualization at the book website, in the section on LXC/LXD.

User namespaces isolate security-related identifiers and attributes of processes, in particular, user IDs and group IDs, the root directory, keys, and very importantly capabilities (as seen in Section W26.9.8). A very distinctive characteristic useful in isolating, or “sandboxing” processes, is that a process's user and group IDs can be different inside and outside a user namespace. In particular, a process can be unprivileged outside a user namespace, but inside have a user ID of 0; in other words, the process has full privileges to utilize system resources inside the user namespace, but cannot have full privileges on system resources outside the namespace. These isolated processes can share the global resources within a locally-scoped, privileged environment.

The details of the namespaces and cgroups API’s are beyond what we can reasonably cover in this chapter. But we do provide an example, which can be found in it’s entirety in the Example sub-section of the user\_namespaces man page on your system. We encourage you to refer to the namespaces, and user\_namespaces man pages on your system for a more complete and detailed description of these two kernel features.

Example W26.28 Namespaces

Objectives: To allow experimentation with establishing various forms of Linux namespaces, using a system program pre-built into the user\_namespaces man page.

Pre-Requisites: Using the man user\_namespaces command, copy the sample program from the Example sub-section, called userns\_child\_exec.c, into a text editor of your choice. Save the file as example\_23\_28.c, and compile it with the command

**gcc -w example\_23\_28.c -o example\_23\_28**

Background: The program example\_23\_28.c is designed to allow experimenting with user namespaces, as well as other types of namespaces. It creates namespaces as specified by command-line options and then executes operations inside those namespaces. Of course, if you are already familiar with the namespaces API, the comments and function inside the program source code provide a basic explanation of the program.

The following Bash session demonstrates its use.

Requirements: Do the steps below, in the order presented, to complete the requirements for this example.

1. Examine the version of the kernel, to verify the level of namespace functionality available on your system. You need to have at least kernel level 3.8 or higher:

$ **uname -rs**

Linux 4.4.0-51-generic

2. Examine your user and group ID numbers, so that you can supply them as subsequent command line arguments:

$ **id -u**

1000

$ **id -g**

1000

3. Create a new user namespace with a new shell(bash), a new user (-U), mount (-m), PID (-p), with user ID (-M) and group ID (-G) 1000 (as found in step 2.), and indexed to PID 0 inside the new namespace. Use the following command:

$ **./example\_23\_28 -p -m -U -M '0 1000 1' -G '0 1000 1' bash**

#

You now have a bash shell in the new namespace, as indicated by the new shell pronpt (#).

4. To see the real and effective user and group id’s in the new namespace, repeat the two commands from step 2.:

# **id -u**

0

# **id -g**

0

You are now a privileged user in this new namespace, since you designated this mapping in the command in step 3. with -M '0 1000 1' and -G '0 1000 1' .

5. The shell in this new namespace is PID 1, as you can see by typing the following command:

# **echo $$**

1

6. Mount a new /proc filesystem, and list all of the processes visible in the new namespace. This shows that the shell is isolated from any processes outside the namespace:

# **mount -t proc proc /proc**

# **ps ax**

PID TTY STAT TIME COMMAND

1 pts/0 S 0:00 bash

38 pts/0 R+ 0:00 ps -ax

7. Inside the user namespace, the shell has user and group ID’s of 0. This is shown by the following command:

namespaces # **cat /proc/$$/status | egrep '^[UG]id'**

Uid: 0 0 0 0

Gid: 0 0 0 0

8. Since we have used the clone() system call in the program example\_23\_28 to clone the user and process ID spaces, any program you run inside the new namespace has a PID assigned inside the namespace. To verify this, open another terminal and use the ps -ax command to find the PID’s of the example\_23\_28 process, and its bash shell. Then compare these PID’s to those inside the new namespace.

9. To terminate the isolated process, type the exit command.

# **exit**

$

Questions and Problems

Questions, Problems, and a Project

W26.1 Write a brief outline of how to install a “server“, non-GUI-interface of your distribution of the Linux operating system, on your computer hardware. Detail exactly how your particular installation procedure differed from a Desktop distribution. If you didn’t do the installation, find out from the system manager how the installation was done, and why it was done in that way.

W26.2 After doing Problem W26.1, do the following steps on the server edition in order to complete the requirements for this problem:

a. If you have not already done so, download, install, and test the vsftpd service on your system, as shown in Example W26.2.

b. Use the **adduser** command, as shown in Example W26.3, to create a new user account on your Linux system with the following configuration-

$ sudo adduser ftp2

Adding user `ftp2' ...

Adding new group `ftp2' (1004) ...

Adding new user `ftp2' (1003) with group `ftp2' ...

Creating home directory `/home/ftp2' ...

Copying files from `/etc/skel' ...

Enter new UNIX password: YYY

Retype new UNIX password: YYY

passwd: password updated successfully

Changing the user information for ftp2

Enter the new value, or press ENTER for the default

Full Name []:

Room Number []:

Work Phone []:

Home Phone []:

Other []:

Is the information correct? [Y/n] Y

$

c. Test your new user account locally on your LAN by using the command **ftp 0** with username **ftp2**. Test it from the Internet. Put files in the users account, and retrieve files from that account locally from another account and from the Internet.

W26.3

After doing Problems W26.1 and w26.2 for a server edition install, and given the steps needed to accomplish user account management shown above in Section 17.3 of the printed book, make a table or chart of what users and groups need to be added to your system, and what their default account parameters and group memberships should be. Then, use the methods shown in Section 17.3 to accomplish user account creation, modification, and deletion from the command line. What command can you use to identify all existing groups on the system? Use the batch mode account creation technique shown in Section 17.3 to implement the users and groups from the table or chart you created.

W26.4 Do the same operations as in Examples W26.8a and b, except use internally-mounted SATA hard disks or USB thumb drives instead of the drive types specified in those Examples. In the case of duplicating Example W26.8a, you will be adding a thumb drive which is ZFS-aware. In the case of duplicating Example W26.8b, you will be doing that on internally-mounted SATA hard drives- you should use three(3) hard drives: the “unmirrored” root, or system disk, and two additional drives you have added after installation of the system.

Recall that Example W26.8c shows the mdadm RAID1 mirroring of the root, or system disk.

Following are some advisories about how to complete the duplicate of Example W26.8a problem successfully:

a. Make sure you know the logical device name of your system disk, such as /dev/sda. Don’t do this problems steps by mistake using that drive! If you do, you will render the system unbootable!

b. Make sure you have installed the zfsutils-linux as shown in the Chapter W22 on ZFS at this book website.

c. You have to unmount the thumb drive before you can use the **zpool** command on it.

d. Create zfs datasets, but be sure to delete the datasets, and destroy the zpool before removing the thumb drive from the machine.

W26.5 Execute Example W26.8c on available hardware. What would be the security and archival advantages of creating an mdadm RAID1, mirrored pair of SATA hard disks, where both ZFS and the root, or system, ext4 file systems are installed on them? To follow up on Example W26.8c, repeat Example W26.8c using a selected higher order of RAID on the requisite number of readily available multiple disks in your array.

W26.6 Add a shared network printer to your Linux system without a direct USB connection, and outline the steps necessary to get the printer to actually work given your installation type.

W26.7 If you haven’t already done so, execute the pre-requisites and the Examples W19.26 through 29 in Chapter W19 on Python at this books website, in Section W19.4.1.1, but only using the Bash and TC shell script code.

W26.8 What is the meaning of the term archive ?

W26.9 What is the tar command used for? Give all its uses.

W26.10 You want to create a tar archive of a project that contains several directories, sub-directories and files, and save the archive on a USB thumbdrive mounted on your system so that you can distribute the archive to your friends. a) What is the pathname to a USB thumbdrive mounted on your system? b)How would you designate a USB thumbdrive as the destination for where the tar archive would be created, as an argument to the tar command? How would you encrypt the USB thumbdrive so that only people with the pass phrase can see what is on it?

W26.11 If you have not encrypted the USB thumbdrive, what are the access permissions for the files on the USB thumbdrive from Problem W26.10 above?

W26.12 Give a command line for creating a tar archive of your current directory.

W26.13 Give commands for compressing and keeping the archive in the backups directory in your home directory.

W26.14 Give commands for restoring the backup file in Problem W26.13 in the ~/backups directory.

W26.15 Give a command line for copying your home directory to a directory called home.back so that access privileges and file modify time are preserved.

W26.16 Why is the tar command preferred over the **cp -r** command for creating backup copies of directory hierarchies?

W26.17 Suppose that you download a file, Linuxbook.tar.Z from an ftp site. Give the sequence of commands for restoring this archive and installing it in your ~/Linuxbook directory.

W26.18 Use the **tar** command to create a compressed archive of a directory of your choosing in a new directory you create named backups under your home directory. Name the compressed archive **something.tar.gz** where something is the name of the directory you chose to backup. Show the command lines that you used to perform these tasks.

W26.19 Use the **tar** command to restore the compressed tar archive ~/backups/courses.tar.gz you produced in Problem W26.18 above, into a new directory named mirrors under your home directory. Show the command lines that you used to perform these tasks.

W26.20 Use the latest version of CloneZilla Live to make a bootable clone of your Linux system disk. The source and target disks for cloning can be either both internally-mounted, or in the case of a laptop computer, internally-mounted for the source, and externally-mounted in a USB or E-SATA enclosure for the target. The instructions for using CloneZilla to do this procedure are found online at the CloneZilla website. Make sure the target disk has a large enough capacity to achieve the cloning!

To test the clone, gracefully shut down your system, and remove the original source system disk. Then replace it with the cloned target, and restart the system.

W26.21 Mansoor is working with Bob on a project. He needs to be able to read, write, create, and delete files related to the project, which are located in the Project directory in Bob's home directory. Bob and Mansoor are ordinary users without administrative privileges. They wish to do this project without contacting the system administrator to request new groups, group membership changes, sudo changes, etc.. When the project is over, Bob will remove the modify permissions on his home and the Project directory for user "mansoor" himself, instead of contacting the system administrator.

On your own Linux system, in conjunction with another user, use ACLs to accomplish the following (substituting valid usernames on your system for Bob and Mansoor):

a. Create a project directory under Bob’s home directory named “Project”.

b. Set the ACL on Bob’s home directory so Mansoor has read, write, and execute privileges on it.

c. Set the ACL on the Project directory so that Mansoor has rwxo privileges on it.

d. Have Bob create some files in the Project directory.

e. Have Mansoor make Bob’s home directory the current directory.

f. Have Mansoor test whether or not he can-

delete files in Bob’s home directory,

delete the Project directory from Bob’s home directory,

list, create new files, or remove the files that Bob put in the Project directory.

g. Have Bob revoke Mansoor’s x privileges on Bob’s home directory and the directory Project.

h. Have Mansoor test the revocation of modify privileges from step g..

i. Why can Mansoor still see the files in Bob’s home directory, and the files in the **Project** directory, but not delete or modify any files in those directories after step g?

j. What chmod command(s) would Bob have to execute to deny Mansoor access to his home directory?

\*Show verification of ACL settings at as many steps as necessary to validate what you have done.

W26.23 If you give a set of users permissions to a project directory using ACLs, how can you ensure that sub-directories that are created by the project manager beneath that project directory provide the same access privileges to those users?

W26.24 Create a project directory on your system and create a git repository in it for any number of local users on your computer system. Then use ACLs to give access to the project directory to the users that are collaborating in the project. This should allow those users to push to and pull from the git repository. Have your allowed users test the repository. Also test the security of the repository, i.e. can non-allowed users access it?

See Chapter W24, Section 5.7 at this books website for more information on creating a git repository.

W26.25 Adding the Universe and Multiverse Repositories- for Debian-family, Linux Mint and Ubuntu systems.

Add one or all of these additional software repositories to either your /etc/apt/sources.list file, or to a separate .list file in the /etc/apt/sources.list.d/ directory

deb http://us.archive.ubuntu.com/ubuntu/ xenial universe

deb-src http://us.archive.ubuntu.com/ubuntu/ xenial universe

deb http://us.archive.ubuntu.com/ubuntu/ xenial-updates universe

deb-src http://us.archive.ubuntu.com/ubuntu/ xenial-updates universe

deb http://us.archive.ubuntu.com/ubuntu/ xenial multiverse

deb-src http://us.archive.ubuntu.com/ubuntu/ xenial multiverse

deb http://us.archive.ubuntu.com/ubuntu/ xenial-updates multiverse

deb-src http://us.archive.ubuntu.com/ubuntu/ xenial-updates multiverse

You should replace 'us.' by another country code, referring to a mirror server in your region.

Don't forget to retrieve the updated package lists with the command sudo apt-get update

W26.26 After completion of Example W26.21, use the Nfs4-acl-tools to accomplish the same things that are done in the example, on a remote NFS filesystem mounted on your local system. In order to do this, you must first install the nfs4-acl-tools on your local system. Then install and use an NFS server on another machine on your network which has an NFS filesystem, and on your local machine install an NFS client. Connect client to server, and mount the remote file system on your machine. Finally, use the Nsf4-acl-tools to set and view ACL inheritance on the files on the remote machine, as is done in Example W26.21.

How do you know ahead of time that the remote NFS filesystem is capable of NFSV4-compliant ACLs?

W26.27 Do all Use Cases from Section W26.9.6.2.1 on your Linux system, and note the results.

W26.28 Use gufw to do all Use Cases in Section W26.9.6.2.1 on your system, and remark on how much easier (or harder) it is to accomplish the same things in gufw.

W26.29 Do all the steps shown in Section W26.9.6.2.4 on your system, and interpret the log entries in the Journal as shown in that section. Use gufw to construct the rule shown in step 1. Furthermore, try different forms of entry, such as vsftp and http on different standard and ephemeral ports, and note the what is present in Journal log entry components for each.

W26.30 Use gufw to create a rule exactly like the deny rule in Section W26.9.6.2.4, step 1.

W26.31

a. Examine the cups.service systemd service unit file in /lib/systemd/system, and write a short paragraph-long description, in your own words, for why CUPS is inactive when no powered-on printers are attached to the system. This question assumes that CUPS is installed on your system, according to the requirements shown in Section W26.5, and that it is enabled at system boot.

b. How would you make CUPS available even though no printer is plugged-in or attached?

W26.32 Execute all of the steps of Example W26.8b up to step 8., using two USB thumb drives mounted on our computer. You can put more data into /mnt/raid1 for testing purposes.

Then, do the following parts:

a. Add a third USB thumb drive equivalent in size to the originals into the RAID1 array md0. On our Linux Mint system, it was named /dev/sdd. Do steps 0. through 3. on the newly added thumb drive.

b. Check the integrity of the data placed in /mnt/raid1.

c. Replace /dev/sdc with the new thumb drive /dev/sdd from part a., and wait for the array to be resynced. On our Linux Mint 18.2 system, this took about 30 minutes.

d. Again, check the integrity of any data you may have placed in /mnt/raid1.

e. Execute step 12. to remove the RAID1 array, and zero out the thumb drives so that they can be reused as FAT32 formatted drives.

W26.33 Execute the steps of the user and group creation method shown in Chapter 17, Section 3.4 in the printed book, in detail for your Linux system, and according to the following constraints:

a. Your systems capabilities to accommodate additional persistent media,

b. whether or not you want to use a ZFS-based or traditional partitioning/file system creation/file system mounting –based approach,

c. your security model, in terms of how it isolates users and groups from one another,

d. your system performance model as it affects users and groups,

e. how many users and groups you plan to accommodate.

For example, you can add one or more additional hard drives to your system in a ZFS-based individual or multiple virtual pools, limit the privileges you want ordinary users to have, allow them to run their own executable programs from their home directories, and have a small-scale set of user accounts and groups.

W26.34 Using the system program shown in Chapter 16, Section 16.5.4 in the printed book, named killer.c, and the general procedure of Example W26.27, give an unprivileged user the capability of terminating any process on your system with the killer.c executable-version program.

Project 1

After completing the Requirements of Example W26.19, automate the Python script file using a systemd timer so that it runs on a schedule that is useful to you.