LFS207

Linux System Administration Essentials

Version 2023-06-30

THE LINUX FOUNDATION | Training & Certification

Version 2023-06-30

Published by:

the **Linux Foundation**
https://www.linuxfoundation.org

Nondisclosure of Confidential Information

"Confidential Information" shall not include any of the following, even if marked confidential or proprietary: (a) information that relates to the code base of any open source or open standards project (collectively, "Open Project"), including any existing or future contribution thereto; (b) information generally relating or pertaining to the formation or operation of any Open Project; or (c) information relating to general business matters involving any Open Project.

This course does not include confidential information, nor should any confidential information be divulged in class.

Contents

Chapter 1

Introduction

1.1 Labs

Exercise 1.1: Configuring the System for sudo

It is very dangerous to run a **root shell** unless absolutely necessary: a single typo or other mistake can cause serious (even fatal) damage.

Thus, the sensible procedure is to configure things such that single commands may be run with superuser privilege, by using the **sudo** mechanism. With **sudo** the user only needs to know their own password and never needs to know the root password.

If you are using a distribution such as **Ubuntu**, you may not need to do this lab to get **sudo** configured properly for the course. However, you should still make sure you understand the procedure.

To check if your system is already configured to let the user account you are using run **sudo**, just do a simple command like:

```
$ sudo ls
```

You should be prompted for your user password and then the command should execute. If instead, you get an error message you need to execute the following procedure.

Launch a root shell by typing **su** and then giving the **root** password, not your user password.

On all recent **Linux** distributions you should navigate to the `/etc/sudoers.d` subdirectory and create a file, usually with the name of the user to whom root wishes to grant **sudo** access. However, this convention is not actually necessary as **sudo** will scan all files in this directory as needed. The file can simply contain:

```
student ALL=(ALL)  ALL
```

if the user is `student`.

An older practice (which certainly still works) is to add such a line at the end of the file `/etc/sudoers`. It is best to do so using the **visudo** program, which is careful about making sure you use the right syntax in your edit.

You probably also need to set proper permissions on the file by typing:

```
$ sudo chmod 440 /etc/sudoers.d/student
```

(Note some **Linux** distributions may require `400` instead of `440` for the permissions.)

After you have done these steps, exit the root shell by typing `exit` and then try to do `sudo ls` again.

There are many other ways an administrator can configure **sudo**, including specifying only certain permissions for certain users, limiting searched paths etc. The `/etc/sudoers` file is very well self-documented.

However, there is one more setting we highly recommend you do, even if your system already has **sudo** configured. Most distributions establish a different path for finding executables for normal users as compared to root users. In particular the directories `/sbin` and `/usr/sbin` are not searched, since **sudo** inherits the `PATH` of the user, not the full root user.

Thus, in this course we would have to be constantly reminding you of the full path to many system administration utilities; any enhancement to security is probably not worth the extra typing and figuring out which directories these programs are in. Consequently, we suggest you add the following line to the `.bashrc` file in your home directory:

```
PATH=$PATH:/usr/sbin:/sbin
```

If you log out and then log in again (you don't have to reboot) this will be fully effective.

Chapter 2

Linux Filesystem Tree Layout

2.1 Labs

Exercise 2.1: Sizes of the Default Linux Directories

Use the **du** utility to calculate the overall size of each of your system's top-level directories.

Type the command:

```
$ du --help
```

for hints on how to obtain and display this result efficiently.

Solution 2.1

To obtain a full list of directories under / along with their size:

```
$ sudo du --max-depth=1 -hx /
```

```
4.3M    /home
16K     /lost+found
39M     /etc
4.0K    /srv
3.6M    /root
178M    /opt
138M    /boot
6.1G    /usr
1.1G    /var
16K     /mnt
4.0K    /media
869M    /tmp
8.4G    /
```

Where we have used the options:

- `--maxdepth=1`: Just go down one level from / and sum up everything recursively underneath in the tree.
- `-h`: Give human-readable numbers (KB, MB, GB).

- `-x` Stay on one filesystem; don't look at directories that are not on the `/` partition. In this case that means ignore:

  `/dev /proc /run /sys`

  because these are pseudo filesystems which exist in memory only; they are just empty mount points when the system is not running. Because this was done on a **RHEL** system, the following mount points are also not followed:

  `/bin /sbin /lib /lib64`

  since they are just symbolically linked to their counterparts under `/usr`.

Exercise 2.2: Touring the /proc Filesystem

Please Note

Exactly what you see in this exercise will depend on your kernel version, so you may not match the output shown precisely.

1. As root, **cd** into `/proc` and do a directory listing. This should display a number of files and directories:

```
$ cd /proc
$ ls -F
```

```
1/      128/    1510/   20/     2411/   30895/  53/    6925/  802/  951/          kmsg
10/     129/    1511/   2015/   2425/   31/     54/    7/     81/   952/          kpagecgroup
1002/   13/     1512/   2022/   2436/   31449/  55/    70/    813/  957/          kpagecount
1007/   130/    1513/   2023/   2444/   32/     56/    702/   814/  97/           kpageflags
10540/  131/    1514/   20300/  2451/   33/     58/    709/   816/  9742/         loadavg
10590/  13172/  152/    20354/  2457/   34/     585/   71/    817/  98/           locks
10798/  132/    15552/  20380/  2489/   35/     59/    718/   82/   99/           meminfo
10805/  133/    15663/  20388/  25/     36/     60/    719/   83/   9923/         misc
10806/  134/    15737/  20392/  2503/   37/     61/    72/    834/  acpi/         modules
10809/  135/    159/    20396/  2504/   374/    6193/  721/   835/  asound/       mounts@
10810/  136/    15981/  2086/   2531/   379/    62/    723/   84/   buddyinfo     mtrr
10813/  137/    16/     2090/   2546/   38/     63/    725/   841/  bus/          net@
10894/  138/    162/    211/    2549/   380/    634/   727/   842/  cgroups       pagetypeinfo
10925/  1384/   1632/   22/     2562/   40/     64/    73/    85/   cmdline       partitions
10932/  1385/   1636/   2205/   25794/  41/     65/    7300/  857/  config.gz     sched_debug
10934/  1387/   166/    2209/   26/     42/     662/   74/    86/   consoles      scsi/
10935/  139/    1670/   2212/   2610/   43/     663/   757/   864/  cpuinfo       self@
10941/  1390/   17/     2232/   26108/  44/     665/   758/   867/  crypto        slabinfo
10983/  1393/   17271/  2238/   2619/   4435/   666/   76/    87/   devices       softirqs
10998/  14/     17361/  2296/   2624/   45/     67/    761/   88/   diskstats     stat
11/     140/    1793/   2298/   2627/   46/     670/   762/   881/  dma           swaps
11047/  1410/   18/     23/     2644/   468/    671/   765/   886/  driver/       sys/
1105/   1415/   1831/   23042/  2645/   47/     673/   766/   887/  execdomains   sysrq-trigger
1121/   1429/   18880/  2344/   2679/   470/    674/   768/   888/  fb            sysvipc/
1123/   1437/   18903/  2348/   27/     484/    678/   769/   889/  filesystems   thread-self@
1135/   1445/   19/     2353/   2706/   49/     679/   77/    89/   fs/           timer_list
11420/  146/    19392/  2354/   2762/   492/    68/    771/   9/    interrupts    timer_stats
11499/  1463/   19488/  2365/   28/     493/    682/   78/    90/   iomem         tty/
11515/  147/    1954/   23683/  2858/   5/      683/   79/    92/   ioports       uptime
11530/  1476/   1963/   2370/   28730/  50/     686/   793/   921/  irq/          version
1163/   148/    19727/  2372/   28734/  51/     687/   794/   928/  kallsyms      vmallocinfo
1164/   1485/   19734/  2374/   29/     510/    69/    8/     930/  kcore         vmstat
12/     149/    19984/  24/     2973/   514/    690/   80/    931/  keys          zoneinfo
127/    15/     2/      2406/   3/      52/     691/   801/   944/  key-users
```

Notice many of the directory names are numbers; each corresponds to a running process and the name is the **process ID**. An important subdirectory we will discuss later is `/proc/sys`, under which many system parameters can be examined or modified.

2. View the following files:

   - /proc/cpuinfo:
   - /proc/meminfo:
   - /proc/mounts:
   - /proc/swaps:
   - /proc/version:
   - /proc/partitions:
   - /proc/interrupts:

   The names give a pretty good idea about what information they reveal.

   Note that this information is not being constantly updated; it is obtained only when one wants to look at it.

3. Take a peek at any random process directory (if it is not a process you own some of the information might be limited unless you use **sudo**):

   ```
   $ ls -F 4435
   ```

   ```
   attr/        coredump_filter  gid_map     mountinfo   oom_score_adj  sessionid  syscall
   autogroup    cpuset           io          mounts      pagemap        setgroups  task/
   auxv         cwd@             limits      mountstats  personality    smaps      timerslack_ns
   cgroup       environ          loginuid    net/        projid_map     stack      uid_map
   clear_refs   exe@             map_files/  ns/         root@          stat       wchan
   cmdline      fd/              maps        oom_adj     sched          statm
   comm         fdinfo/          mem         oom_score   schedstat      status
   ```

   Take a look at some of the fields in here such as: `cmdline`, `cwd`, `environ`, `mem`, and `status`

Chapter 3

User Environment

3.1 Labs

Exercise 3.1: Adding the ~/work directory to your path

Create a small file, ~/work/ls, which contains just the line:

```
echo HELLO, this is the phony ls program.
```

Then make it executable by doing:

```
$ chmod +x ~/work/ls
```

1. Append ~/work to your path, so it is searched only **after** your usual path is considered. Type **ls** and see which program is run: /bin/ls or ~/work/ls.
2. Pre-pend ~/work to your path, so it is searched **before** your usual path is considered. Once again, type **ls** and see which program is run: /bin/ls or ~/work/ls.

What are the security considerations in altering the path this way?

Solution 3.1

First create the phony **ls** program, using an editor, or just simply doing:

```
$ echo "echo HELLO, this is the phony ls program." > ~/work/ls
$ chmod +x ~/work/ls
```

For the next two steps it is a good idea to work in another terminal window, or just start a new shell, so the changes do not persist on later issued commands. You can start a new shell by just typing **bash**.

1.
   ```
   $ bash
   $ PATH=$PATH:~/work
   ```

   ```
   bin  etc  games  include  lib  lib64  libexec  local  sbin  share  src        tmp
   ```

   ```
   $ exit
   ```

2.
```
$ bash
$ PATH=~/work:$PATH
$ ls /usr
```
```
HELLO, this is the phony ls program.
```
```
$ exit
```

Very Important

Note the second form is a very dangerous thing to do, and is a trivial way to insert a **Trojan Horse** program; if someone can put a malicious program in `~/work`, they can trick you into running it accidentally.
Make sure other users don't have **write** permission to directories in your `\$PATH`.

Exercise 3.2: Command history

You have been busy working with at your **Linux** workstation long enough to have typed in about 100 commands in one particular **bash** command shell.

At some point earlier, you used a new command, but the exact name has slipped your mind.

Or perhaps it was a pretty complicated command with a bunch of options and arguments and you don't want to go through the error prone process of figuring out what to type again.

How do you ascertain what the command was?

Once you find the command in your history, how do you easily issue the command again without having to type it all in at the prompt?

Solution 3.2

Command history:

The **history** command is the way to display the commands you have typed in:

```
$ history
```
```
    1  cd /
    2  ls
    3  cd
    4  pwd
    5  echo $SHELL
    6  ls /var/
    7  ls /usr/bin
    8  ls /usr/local/bin
    9  man fstab
   10  ls
            . . .
```

In order to re-run a previous command, you have a few choices. Let's say that you wanted to re-run the **man** command you ran way back when you first logged-in. You could type

```
$ !9
```

to re-run the command listed as `#9`. If this was the only **man** command that you typed in, you could also type

```
$ !man
```

now that you remember the command name that you typed. Finally, if you had typed a few **man** commands, you could use `CTRL-R` to search backward in your history to find the specific **man** command that you want to re-run, and then just hit `Return` to execute it.

Exercise 3.3: Command alias

Typing long commands and filenames over and over again gets rather tedious, and leads to a lot of trivial errors such as typos.

Deploying **aliases** allows us to define short cuts to alleviate the pain of all of this typing.

Suppose you are a member of a project team that works in a common, shared directory for your project. This directory is located in `/home/staff/RandD/projects/projectX/src`.

When you are working on `Project X`, you often need to create and modify your files in this directory. It doesn't take too long before typing in:

```
$ cd /home/staff/RandD/projects/projectX/src
```

gets tedious.

Define and use an alias named "`projx`" to do the above **cd** command for you.

Solution 3.3

Command alias:

The **alias** line would look like this:

```
$ alias projx='cd /home/staff/RandD/projects/projectX/src'
```

Note you can use double quotes instead of single quotes, or use no quotes at all since their is no white space in your defined alias.

All you have to do now to change to the directory is

```
$ projx
```

To make the alias persistent, just place it in your `$HOME/.bashrc` file.

Chapter 4

User Account Management

4.1 Labs

Exercise 4.1: Accounts

Which account are you logged-in as? How do you find out?

Solution 4.1 Run the **id** utility:

```
$ id
```

```
uid=1000(user) gid=1000(user) groups=1000(user),999(qubes)
```

Please Note

There are many ways to accomplish this. Here are some others:
$ whoami
$ grep ^Uid /proc/$$/status

Exercise 4.2: Working with User Accounts

1. Examine `/etc/passwd` and `/etc/shadow`, comparing the fields in each file, especially for the normal user account. What is the same and what is different?
2. Create a `user1` account using **useradd**.
3. Login as `user1` using `ssh`. You can just do this with:

   ```
   $ ssh user1@localhost
   ```

   It should fail because you need a password for `user1`; it was never established.
4. Set the password for `user1` to `user1pw` and then try to login again as `user1`.
5. Look at the new records which were created in the `/etc/passwd`, `/etc/group` and the `/etc/shadow` files.

6. Look at the `/etc/default/useradd` file and see what the current defaults are set to. Also look at the `/etc/login.defs` file.
7. Create a user account for `user2` which will use the **Korn** shell (**ksh**) as its default shell. (if you do not have /**bin**/**ksh** install it or use the **C** shell at /**bin**/**csh**.) Set the password to `user2pw`.
8. Look at `/etc/shadow`. What is the current expiration date for the `user1` account?
9. Use **chage** to set the account expiration date of `user1` to December 1, 2013.

   Look at `/etc/shadow` to see what the new expiration date is.
10. Use **usermod** to lock the `user1` account.

    Look at `/etc/shadow` and see what has changed about `user1`'s password. Reset the password to `userp1` on the account to complete this exercise.

Solution 4.2

1. `$ sudo grep student /etc/passwd /etc/shadow`

```
/etc/passwd:student:x:1000:100:LF Student:/home/student:/bin/bash
/etc/shadow:student:$6$jtoFVPICHhba$iGFFUU8ctrtOGoistJ4/30DrNLi1FS66qnn0VbS6Mvm
  luKI08SgbzT5.IcOHo5j/SOdCagZmF2RgzTvzLb11H0:16028:0:99999:7:::
```

   (You can use any normal user name in the place of `student`.) About the only thing that matches is the user name field.
2. `$ sudo useradd user1`
3. `$ ssh user1@localhost`

```
user1@localhost's password:
```

   Note you may have to first start up the **sshd** service as in:

   `$ sudo service sshd restart`

   or

   `$ sudo systemctl restart sshd.service`
4. `$ sudo passwd user1`

```
Changing password for user user1.
New password:
```

5. `$ sudo grep user1 /etc/passwd /etc/shadow`

```
/etc/passwd:user1:x:1001:100::/home/user1:/bin/bash
/etc/shadow:user1:$6$OBE1mPMw$CIc7urbQ9ZSnyiniV0eJxKqLFu8fz4whfEexVem2
   TFpucuwRN1CCHZ19XGhj4qVujslRIS.P4aCXd/y1U4utv.:16372:0:99999:7:::
```

6. On either **RHEL** or **openSUSE** systems for example:

   `$ cat /etc/default/useradd`

```
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

```
$ cat /etc/login.defs
```

```
....
```

We don't reproduce the second file as it is rather longer, but examine it on your system.

7.
```
$ sudo useradd -s /bin/ksh user2
$ sudo passwd user2
```

```
Changing password for user user2.
New password:
```

8.
```
$ sudo grep user1 /etc/shadow
```

```
user1:$6$OBE1mPMw$CIc7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2TFpucuwRN1CCHZ
   19XGhj4qVujslRIS.P4aCXd/y1U4utv.:16372:0:99999:7:::
```

There should be no expiration date.

9.
```
$ sudo chage -E 2013-12-1 user1
$ sudo grep user1 /etc/shadow
```

```
  user1:$6$OBE1mPMw$CIc7urbQ9ZSnyiniVOeJxKqLFu8fz4whfEexVem2TFpucuwRN1CCHZ
19XGhj4qVujslRIS.P4aCXd/y1U4utv.:16372:0:99999:7::16040:
```

10.
```
$ sudo usermod -L user1

$ sudo passwd user1
```

Chapter 5

Group Management

5.1 Labs

Exercise 5.1: Working with Groups

1. Create two new user accounts (`rocky` and `bullwinkle` in the below) and make sure they have home directories.
2. Create two new groups, `friends` and `bosses` (with a GID of 490). Look at `/etc/group`. See what GID was given to each new group.
3. Add `rocky` to both new groups.

   Add `bullwinkle` to group `friends`.

   Look in `/etc/group` to see how it changed.
4. Login as `rocky`. Create a directory called `somedir` and set the group ownership to `bosses`. (Using **chgrp** which will be discussed in the next session.)

   (You will probably need to add execute privileges for all on `rocky`'s home directory.)
5. Login as `bullwinkle` and try to create a file in `/home/rocky/somedir` called `somefile` using the **touch** command.

   Can you do this? No, because of the group ownership and the `chmod a+x` on the directory.
6. Add `bullwinkle` to the `bosses` group and try again. Note you will have to log out and log back in again for the new group membership to be effective. Do the following:

Solution 5.1

1.
   ```
   $ sudo useradd -m rocky
   $ sudo useradd -m bullwinkle
   $ sudo passwd rocky
   ```

   ```
   Enter new UNIX password:
   Retype new UNIX password:
   passwd: password updated successfully
   ```

   ```
   $ sudo passwd bullwinkle
   ```

```
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

```
$ ls -l /home
```

```
total 12
drwxr-xr-x  2 bullwinkle bullwinkle 4096 Oct 30 09:39 bullwinkle
drwxr-xr-x  2 rocky      rocky      4096 Oct 30 09:39 rocky
drwxr-xr-x 20 student    student    4096 Oct 30 09:18 student
```

2.
```
$ sudo groupadd friends
$ sudo groupadd -g 490 bosses
$ grep -e friends -e bosses /etc/group
```

```
friends:x:1003:
bosses:x:490:
```

3.
```
$ sudo usermod -G friends,bosses rocky
$ sudo usermod -G friends bullwinkle

$ grep -e rocky -e bullwinkle /etc/group
```

```
rocky:x:1001:
bullwinkle:x:1002:
friends:x:1003:rocky,bullwinkle
bosses:x:490:rocky
```

```
$ groups rocky bullwinkle
```

```
rocky : rocky friends bosses
bullwinkle : bullwinkle friends
```

4.
```
$ ssh rocky@localhost
$ cd ~
$ mkdir somedir
$ chgrp bosses somedir
$ ls -l
```

```
total 16
-rw-r--r-- 1 rocky rocky  8980 Oct  4  2013 examples.desktop
drwxrwxr-x 2 rocky bosses 4096 Oct 30 09:53 somedir
```

```
$ chmod a+x .
```

5.
```
$ ssh bullwinkle@localhost
$ touch /home/rocky/somedir/somefile
```

```
touch: cannot touch /home/rocky/somedir/somefile: Permission denied
```

```
$ exit
```

6.
```
$ sudo usermod -a -G bosses bullwinkle
$ ssh bullwinkle@localhost
$ touch /home/rocky/somedir/somefile
$ ls -al /home/rocky/somedir
```

(note ownership of files)

Chapter 6

File Permissions and Ownership

6.1 Labs

Exercise 6.1: Using chmod

One can use either the octal digit or symbolic methods for specifying permissions when using **chmod**. Let's elaborate some more on the symbolic method.

It is possible to either give permissions directly, or add or subtract permissions.

Try the following examples:

```
$ chmod u=r,g=w,o=x afile
$ chmod u+w,g-w,o+rw afile
$ chmod ug=rwx,o-rw afile
```

After each step do:

```
$ ls -l afile
```

to see how the permissions took, and try some variations.

Exercise 6.2: umask

Create an empty file with:

```
$ touch afile
$ ls -l afile
```

```
-rw-rw-r-- 1 coop coop 0 Jul 26 12:43 afile
```

which shows it is created by default with both read and write permissions for owner and group, but only read for world.

In fact, at the operating system level the default permissions given when creating a file or directory are actually read/write for owner, group **and** world (0666); the default values have actually been modified by the current **umask**.

If you just type **umask** you get the current value:

```
$ umask
```

```
0002
```

which is the most conventional value set by system administrators for users. This value is combined with the file creation permissions to get the actual result; i.e.,

```
0666 & ~002 = 0664; i.e., rw-rw-r--
```

Try modifying the **umask** and creating new files and see the resulting permissions, as in:

```
$ umask 0022
$ touch afile2
$ umask 0666
$ touch afile3
$ ls -l afile*
```

Exercise 6.3: Using Access Control Lists

1. Create a file using your usual user name and run **getfacl** on it to see its properties.
2. Create a new user account with default properties (or reuse one from previous exercises).
3. Login as that user and try to add a line to the file you created in the first step. This should fail.
4. Use **setfacl** to make the file writeable by the new user and try again.
5. Use **setfacl** to make the file not readable by the new user and try again.
6. Clean up as necessary.

Solution 6.3

It is probably easiest to open two terminal windows, one to work in as your normal user account, and the other as the secondary one.

1. In window 1:

```
$ echo This is a file > /tmp/afile
$ getfacl /tmp/afile
```

```
getfacl: Removing leading '/' from absolute path names
# file: tmp/afile
# owner: coop
# group: coop
user::rw-
group::rw-
other::r--
```

2. In window 1:

```
$ sudo useradd fool
$ sudo passwd fool
...
```

3. In window 2:

```
$ sudo su - fool
$ echo another line > /tmp/afile
```

```
-bash: /tmp/afile: Permission denied
```

4. In window 1:

```
$ setfacl -m u:fool:rw /tmp/afile
$ getfacl /tmp/afile
```

```
getfacl: Removing leading '/' from absolute path names
# file: tmp/afile
# owner: coop
# group: coop
user::rw-
user:fool:rw-
group::rw-
mask::rwx
other::r--
```

In window 2:

```
$ echo another line > /tmp/afile
```

5. In window 1:

```
$ setfacl -m u:fool:w /tmp/afile
```

In window 2:

```
$ echo another line > /tmp/afile
-bash: /tmp/afile: Permission denied
```

6. Cleaning up:

```
$ rm /tmp/afile
$ sudo userdel -r fool
```

Chapter 7

Package Management Systems

7.1 Labs

Exercise 7.1: Version Control with git

We will have more detailed discussion of **git** in later sections. Here, we will just get a basic feel for how to use it.

Making sure git is installed

Your system may already have **git** installed. Doing `which git` should show you if it is already present. If not, while you may obtain the source and compile and install it, it is usually easier to install the appropriate pre-compiled binary packages. Exact package names may vary, but one of the following should work depending on your distribution:

```
$ sudo apt install git*      # Debian /Ubuntu
$ sudo zypper install git*   # openSUSE
$ sudo dnf install git*      # Fedora / RHEL 8 / CentOS 8
$ sudo yum install git*      # RHEL 7 / CentOS 7
```

according to your particular distribution.

Let's get a feel for how **git works** and how easy it easy to use. For now we will just make our own local project.

1. First we create a working directory and then initialize **git** to work with it:

   ```
   $ mkdir git-test
   $ cd git-test
   $ git init
   ```

2. Initializing the project creates a `.git` directory which will contain all the version control information; the main directories included in the project remain untouched. The initial contents of this directory look like:

   ```
   $ ls -l .git
   ```

   ```
   total 40
   drwxrwxr-x 2 coop coop 4096 Dec 30 13:59 branches/
   ```

```
-rw-rw-r-- 1 coop coop   92 Dec 30 13:59 config
-rw-rw-r-- 1 coop coop   58 Dec 30 13:59 description
-rw-rw-r-- 1 coop coop   23 Dec 30 13:59 HEAD
drwxrwxr-x 2 coop coop 4096 Dec 30 13:59 hooks/
drwxrwxr-x 2 coop coop 4096 Dec 30 13:59 info/
drwxrwxr-x 4 coop coop 4096 Dec 30 13:59 objects/
drwxrwxr-x 4 coop coop 4096 Dec 30 13:59 refs/
```

Later we will describe the contents of this directory and its subdirectories; for the most part they start out empty.

3. Next we create a file and add it to the project:

```
$ echo some junk > somejunkfile
$ git add somejunkfile
```

4. We can see the current status of our project with:

```
$ git status
```

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

      new file: somejunkfile
```

Notice it is telling us that our file is **staged** but not yet **committed**.

5. Let's tell **git** who is responsible for this repository:

```
$ git config user.name "Another Genius"
$ git config user.email "b_genius@linux.com"
```

This must be done for each new project unless you have it predefined in a global configuration file.

6. Now let's modify the file, and then see the history of differences:

```
$ echo another line >> somejunkfile
$ git diff
```

```
diff --git a/somejunkfile b/somejunkfile
index 9638122..6023331 100644
--- a/somejunkfile
+++ b/somejunkfile
@@ -1 +1,2 @@
 some junk
+another line
```

7. To actually commit the changes to the repository we do:

```
$  git commit -m "My initial commit"
```

```
Created initial commit eafad66: My initial commit
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 somejunkfile
```

If you do not specify an identifying message to accompany the commit with the `-m` option you will jump into an editor to put some content in. You **must** do this or the commit will be rejected. The editor chosen will be what is set in your `EDITOR` environment variable, which can be superseded with setting `GIT_EDITOR`.

8. You can see your history with:

   ```
   $ git log
   ```

   ```
   commit eafad66304ebbcd6acfe69843d246de3d8f6b9cc
   Author: A Genius <a_genius@linux.com>
   Date:   Wed Dec 30 11:07:19 2009 -0600

       My initial commit
   ```

   and you can see the information got in there. You will note the long hexadecimal string which is the **commit number**; it is a 160-bit, 40-digit unique identifier. **git** cares about these beasts, not file names.

9. You are now free to modify the already exiting file and add new files with `git add`. But they are staged until you do another `git commit`

10. Now that was not so bad. But we have only scratched the surface.

Chapter 8

dpkg

8.1 Labs

Exercise 8.0: dpkg Required

On Debian, Ubuntu, or Linux Mint

To do these labs you need to have access to a system that is **Debian**-based, such as **Debian**, **Ubuntu**, or **Linux Mint**.

Exercise 8.1: Using dpkg

Here we will just do a number of simple operations for querying and verifying **Debian** packages.

1. Find out what package the file `/etc/logrotate.conf` belongs to.
2. List information about the package including all the files it contains.
3. Verify the package installation.
4. Try to remove the package.

Solution 8.1

1. `$ dpkg -S logrotate.conf`

```
logrotate: /usr/share/man/man5/logrotate.conf.5.gz
logrotate: /etc/logrotate.conf
rsync: /usr/share/doc/rsync/examples/logrotate.conf.rsync
```

2. `$ dpkg -L logrotate`

```
...
```

3. `$ dpkg -V logrotate`

4. `$ sudo dpkg -r logrotate`

```
dpkg: dependency problems prevent removal of logrotate:
 ubuntu-standard depends on logrotate; however:
  Package logrotate is to be removed.
 libvirt-daemon-system depends on logrotate.

dpkg: error processing package logrotate (--remove):
 dependency problems - not removing
Errors were encountered while processing:
 logrotate
```

Chapter 9

APT

9.1 Labs

Exercise 9.0: dpkg Required

On Debian, Ubuntu, or Linux Mint

To do these labs you need to have access to a system that is **Debian**-based, such as **Debian**, **Ubuntu**, or **Linux Mint**.

Exercise 9.1: Basic APT Commands

1. Check to see if there are any available updates for your system.
2. List all installed kernel-related packages, and list all installed or available ones.
3. Install the **apache2-dev** package, or anything else you might not have installed yet. Doing a simple:

   ```
   $ apt-cache pkgnames
   ```

   will let you see a complete list; you may want to give a wildcard argument to narrow the list.

Solution 9.1

1. First synchronize the package index files with remote repositories:

   ```
   $ sudo apt update
   ```

   To actually upgrade:

   ```
   $ sudo apt upgrade
   $ sudo apt -u upgrade
   ```

   (You can also use `dist-upgrade` as discussed earlier.) Only the first form will try to do the installations.

2.
   ```
   $ apt-cache search "kernel"
   $ apt-cache search -n "kernel"
   $ apt-cache pkgnames "kernel"
   ```

The second and third forms only find packages that have `kernel` in their name.

```
$ dpkg --get-selections "*kernel*"
```

to get only installed packages. Note that on **Debian**-based systems you probably should use `linux` not `kernel` for kernel-related packages as they don't usually have `kernel` in their name.

3. `$ sudo apt install apache2-dev`

Exercise 9.2: Using APT to Find Information About a Package

Using **apt-cache** and **apt** (and not **dpkg**), find:

1. All packages that contain a reference to **bash** in their name or description.
2. Installed and available **bash** packages.
3. The package information for **bash.**
4. The dependencies for the **bash** package.

Try the commands you used above both as `root` and as a regular user. Do you notice any difference?

Solution 9.2

1. `$ apt-cache search bash`
2. `$ apt-cache search -n bash`
3. `$ apt-cache show bash`
4.
```
$ apt-cache depends bash
$ apt-cache rdepends bash
```

Exercise 9.3: Managing Groups of Packages with APT

APT provides the ability to manage groups of packages, similarly to the way **dnf** does it, through the use of **metapackages**. These can be thought of as **virtual packages**, that collect related packages that must be installed and removed as a group.

To get a list of of available **metapackages**:

```
$ apt-cache search metapackage
```

```
bacula - network backup service - metapackage
bacula-client - network backup service - client metapackage
bacula-server - network backup service - server metapackage
cloud-utils - metapackage for installation of upstream cloud-utils source
compiz - OpenGL window and compositing manager
emacs - GNU Emacs editor (metapackage)
....
```

You can then easily install them like regular single packages, as in:

```
$ sudo apt install bacula-client
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  bacula-common bacula-console bacula-fd bacula-traymonitor
```

```
Suggested packages:
  bacula-doc kde gnome-desktop-environment
The following NEW packages will be installed:
  bacula-client bacula-common bacula-console bacula-fd bacula-traymonitor
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 742 kB of archives.
After this operation, 1,965 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Select an uninstalled metapackage and then remove it.

Chapter 10

RPM

10.1 Labs

Exercise 10.0: RPM Required

On Red Hat, Centos, Fedora, or openSUSE

To do these labs you need to have access to a system that is **RPM**-based, such as **RHEL**, **CentOS**, **Fedora**, **SUSE**, or **openSUSE**.

Exercise 10.1: Using RPM

Here we will just do a number of simple operations for querying and verifying **rpm** packages.

This lab will work equally well on **Red Hat** and **SUSE**-based systems.

1. Find out what package the file `/etc/logrotate.conf` belongs to.
2. List information about the package including all the files it contains.
3. Verify the package installation.
4. Try to remove the package.

Solution 10.1

1. `$ rpm -qf /etc/logrotate.conf`

```
logrotate-3.14.0-3.el8.x86_64
```

2. `$ rpm -qil logrotate`

```
...
```

Note a fancier form that combines these two steps would be:

```
$ rpm -qil $(rpm -qf /etc/logrotate.conf)
```

3. `$ rpm -V logrotate`

```
..?......    /etc/cron.daily/logrotate
S.5....T.  c /etc/logrotate.conf
```

On Red Hat

4. On **RHEL 8**:
   `$ sudo rpm -e logrotate`

```
error: Failed dependencies:
        logrotate is needed by (installed) vsftpd-3.0.3-28.el8.x86_64
        logrotate >= 3.5.2 is needed by (installed) rsyslog-8.37.0-13.el8.x86_64
```

On openSUSE

On **openSUSE-Leap 15.1**:
`$ sudo  rpm -e logrotate`

```
error: Failed dependencies:
        logrotate is needed by (installed) xdm-1.1.11-lp151.13.2.x86_64
        logrotate is needed by (installed) wpa_supplicant-2.6-lp151.4.4.x86_64
        logrotate is needed by (installed) chrony-3.2-lp151.8.6.x86_64
        logrotate is needed by (installed) net-snmp-5.7.3-lp151.7.5.x86_64
        logrotate is needed by (installed) syslog-service-2.0-lp151.3.3.noarch
        logrotate is needed by (installed) vsftpd-3.0.3-lp151.6.3.x86_64
        logrotate is needed by (installed) libvirt-daemon-5.1.0-lp151.7.6.1.x86_64
        logrotate is needed by (installed) iscsiuio-0.7.8.2-lp151.13.6.1.x86_64
        logrotate is needed by (installed) mcelog-1.60-lp151.2.3.1.x86_64
```

Note that the exact package dependency tree depends on both the distribution and choice of installed software.

Exercise 10.2: Rebuilding the RPM Database

There are conditions under which the **RPM** database stored in `/var/lib/rpm` can be corrupted. In this exercise we will construct a new one and verify its integrity.

This lab will work equally well on **Red Hat** and **SUSE**-based systems.

1. Backup the contents of `/var/lib/rpm` as the rebuild process will overwrite the contents. If you neglect to do this and something goes wrong you are in serious trouble.
2. Rebuild the data base.
3. Compare the new contents of the directory with the backed up contents; don't examine the actual file contents as they are binary data, but note the number and names of the files.
4. Get a listing of all **rpms** on the system. You may want to compare this list with one generated before you actually do the rebuild procedure. If the query command worked, your new database files should be fine.
5. Compare again the two directory contents. Do they have the same files now?
6. You could delete the backup (probably about 100 MB in size) but you may want to keep it around for a while to make sure your system is behaving properly before trashing it.

✓ Solution 10.2

1.
   ```
   $ cd /var/lib
   $ sudo cp -a rpm rpm_BACKUP
   ```

2.
   ```
   $ sudo rpm --rebuilddb
   ```

3.
   ```
   $ ls -l rpm rpm_BACKUP
   ```

4.
   ```
   $ rpm -qa | tee /tmp/rpm-qa.output
   ```

5.
   ```
   $ ls -l rpm rpm_BACKUP
   ```

6. Probably you should not do this until you are sure the system is fine!

   ```
   $ sudo rm -rf rpm_BACKUP
   ```

Chapter 11

dnf and yum

11.1 Labs

Exercise 11.0: dnf or yum Required

On Red Hat, Centos, or Fedora

To do these labs you need to have access to a system that is **dnf**-based, such as **RHEL**/**CentOS** or **Fedora**. On **RHEL**/**CentOS 7** you should be able to substitute **yum** for **dnf** in the following commands and solution suggestions, as **dnf** is backwards compatible.

Exercise 11.1: Basic dnf Commands

1. Check to see if there are any available updates for your system.
2. Update a particular package.
3. List all installed kernel-related packages, and list all installed or available ones.
4. Install the **httpd-devel** package, or anything else you might not have installed yet. Doing a simple:

   ```
   $ sudo dnf list
   ```

   will let you see a complete list; you may want to give a wildcard argument to narrow the list.

Solution 11.1

1.
   ```
   $ sudo dnf update
   $ sudo dnf check-update
   $ sudo dnf list updates
   ```

   Only the first form will try to do the installations.

2.
   ```
   $ sudo dnf update bash
   ```

3.
```
$ sudo dnf list installed "kernel*"
$ sudo dnf list "kernel*"
```

4.
```
$ sudo dnf install httpd-devel
```

Exercise 11.2: Finding Information About a Package

Using **dnf** (and not **rpm** directly), find:

1. All packages that contain a reference to **bash** in their name or description.
2. Installed and available **bash** packages.
3. The package information for **bash**.
4. The dependencies for the **bash** package.

Try the commands you used above both as `root` and as a regular user. Do you notice any difference?

Solution 11.2

Please Note

Depending on your distribution version, you may get some permission errors if you do not use **sudo** with the following commands, even though we are just getting information.

```
$ sudo dnf search bash
$ sudo dnf list bash
$ sudo dnf info bash
$ sudo dnf deplist bash
```

Exercise 11.3: Managing Groups of Packages

Please Note

On **RHEL/CentOS** you may get some permission errors if you don't use **sudo** with some of the following commands, even when we are just getting information.

dnf provides the ability to manage groups of packages.

1. Use the following command to list all package groups available on your system:

   ```
   $ dnf grouplist
   ```

2. Identify the `Virtualization Host` group and generate the information about this group using the command

   ```
   $ dnf groupinfo "Virtualization Host"
   ```

3. Install using:

   ```
   $ sudo dnf groupinstall "Virtualization Host"
   ```

4. Identify a package group that's currently installed on your system and that you don't need. Remove it using `dnf groupremove` as in:

   ```
   $ sudo dnf groupremove "Virtualization Host"
   ```

Note you will be prompted to confirm removal so you can safely type the command to see how it works.

You may find that the `groupremove` does **not** remove everything that was installed; whether this is a bug or a feature can be discussed.

Exercise 11.4: Adding a New Repository

According to its authors (at http://www.webmin.com/index.htm):

"**Webmin** is a web-based interface for system administration for Unix. Using any modern web browser, you can setup user accounts, Apache, DNS, file sharing and much more. Webmin removes the need to manually edit Unix configuration files like /etc/passwd, and lets you manage a system from the console or remotely."

We are going to create a repository for installation and upgrade. While we could simply go to the download page and get the current **rpm**, that would not automatically give us any upgrades.

1. Create a new repository file called `webmin.repo` in the `/etc/yum.repos.d` directory. It should contain the following:

   webmin.repo

   ```
   [Webmin]
   name=Webmin Distribution Neutral
   baseurl=http://download.webmin.com/download/yum
   mirrorlist=http://download.webmin.com/download/yum/mirrorlist
   enabled=1
   gpgcheck=0
   ```

2. Install the webmin package.

   ```
   $ sudo dnf install webmin
   ```

Chapter 12

zypper

12.1 Labs

Exercise 12.0: zypper Required

On openSUSE

To do these labs you need to have access to a system that is **zypper**-based, such as **SUSE**, or **openSUSE**.

Exercise 12.1: Basic zypper Commands

1. Check to see if there are any available updates for your system.
2. Update a particular package.
3. List all repositories the system is aware of, enabled or not.
4. List all installed kernel-related packages, and list all installed or available ones.
5. Install the **apache2-devel** package, or anything else you might not have installed yet. (Note **httpd** is **apache2** on **SUSE** systems.) Doing a simple:

   ```
   $ sudo zypper search
   ```

   will let you see a complete list; you may want to give a wildcard argument to narrow the list.

Solution 12.1

1. `$ zypper list-updates`
2. `$ sudo zypper update bash`
3. `$ zypper repos`

4.
   ```
   $ zypper search -i kernel
   $ zypper search kernel
   ```

5.
   ```
   $ sudo zypper install apache2-devel
   ```

Exercise 12.2: Using zypper to Find Information About a Package

Using **zypper** (and not **rpm** directly), find:

1. All packages that contain a reference to **bash** in their name or description.
2. Installed and available **bash** packages.
3. The package information for **bash**.
4. The dependencies for the **bash** package.

Try the commands you used above both as `root` and as a regular user. Do you notice any difference?

Solution 12.2

1.
   ```
   $ zypper search -d bash
   ```

   Without the `-d` option only packages with `bash` in their actual name are reported. You may have to do `zypper info` on the package to see where **bash** is mentioned.

2.
   ```
   $ zypper search bash
   ```

3.
   ```
   $ zypper info  bash
   ```

4.
   ```
   $ zypper info --requires  bash
   ```

   will give a list of files **bash** requires. Perhaps the easiest way to see what depends on having **bash** installed is to do

   ```
   $ sudo zypper remove --dry-run bash
   ```

   For this exercise **bash** is a bad choice since it is so integral to the system; you really can't remove it anyway.

Chapter 13

GIT Fundamentals

13.1 Labs

Exercise 13.1: Clone an upstream software repository and make local changes

We are going to walk through a relatively common task in systems management – dealing with a software package that our distribution either doesn't have packaged or its the version is out of date.

The package we will use is *moreutils* from http://joeyh.name/code/moreutils/, which **is** probably packaged by your distribution and up to date, but this example:

1. has a fair number of **branches** and **tags** to look at
2. doesn't require a large number of dependencies like many great `Rust` or `Go` utilities.
3. is actually useful if you want to install it and keep it around

We will target the **Debian** version to be a little nicer to the source server and to allow for a bit more repository information. Use this url for the exercise:

https://salsa.debian.org/nsc/moreutils

1. First **clone** the repository.
2. Create a local branch called `local/latest` based off the `debian/latest` branch.
3. Create a file called `README.local` inside the repository with anything you want.
4. Commit your new file on the new branch. Refer to the previous section **Minimum Global Config** if you need to set up a `user.name` and `user.email`
5. (Optional) Create an annotated tag based on the current version.
6. (Optional) Build the package and experiment with the tools.

At various stages of doing all this, examine the contents of the `.git` directory and see what files are being changed, what their content is, etc. Learn as much as you can.

Solution 13.1

1. `$ git clone https://salsa.debian.org/nsc/moreutils`

```
$ git clone https://salsa.debian.org/nsc/moreutils
Cloning into 'moreutils'...
warning: redirecting to https://salsa.debian.org/nsc/moreutils.git/
remote: Enumerating objects: 2107, done.
remote: Counting objects: 100% (229/229), done.
remote: Compressing objects: 100% (105/105), done.
remote: Total 2107 (delta 129), reused 208 (delta 121), pack-reused 1878
Receiving objects: 100% (2107/2107), 510.79 KiB | 1.45 MiB/s, done.
Resolving deltas: 100% (1349/1349), done.
```

2. `$ git checkout debian/latest`

```
Already on 'debian/latest'
Your branch is up to date with 'origin/debian/latest'.
```

(OR)

`$ git status`

```
On branch debian/latest
Your branch is up to date with 'origin/debian/latest'.

nothing to commit, working tree clean
```

This was actually a **NOOP** because the **default branch** is already `debian/latest`.
So either of these commands will work:

`$ git checkout -b local/latest`

```
Switched to a new branch 'local/latest'
```

(OR)

`$ git checkout -b local/latest debian/latest`

```
Switched to a new branch 'local/latest'
```

But the last one will work regardless of which textbfbranch you currently have checked out.

3. `$ touch README.local`

4. `$ git add README.local`
   `$ git commit README.local`

```
[local/latest cc7d808] add local Tracking file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.local
```

5. `$ git tag -a vlocal/1.0 -m 'first local version'`

Chapter 14

Processes

14.1 Labs

Exercise 14.1: Controlling Processes with ulimit

Please do:

```
$ help ulimit
```

and read `/etc/security/limits.conf` before doing the following steps.

1. Start a new shell by typing **bash** (or opening a new terminal) so that your changes are only effective in the new shell. View the current limit on the number of open files and explicitly view the hard and soft limits.
2. Set the limit to the hard limit value and verify if it worked.
3. Set the hard limit to 2048 and verify it worked.
4. Try to set the limit back to the previous value. Did it work?

Solution 14.1

1.
   ```
   $ bash
   $ ulimit -n
   ```
   ```
   1024
   ```
   ```
   $ ulimit -S -n
   ```
   ```
   1024
   ```
   ```
   $ ulimit -H -n
   ```
   ```
   4096
   ```

2.
   ```
   $ ulimit -n hard
   $ ulimit -n
   ```

```
4096
```

3. `$ ulimit -n 2048`
   `$ ulimit -n`

```
2048
```

4. `$ ulimit -n 4096`

```
bash: ulimit: open files: cannot modify limit: Operation not permitted
```

   `$ ulimit -n`

```
2048
```

   You can't do this anymore!

   Note that if we had chosen a different limit, such as stack size (`-s`) we could raise back up again as the hard limit is `unlimited`.

Exercise 14.2: Examining System V IPC Activity

System V IPC is a rather old method of **I**nter **P**rocess **C**ommunication that dates back to the early days of **UNIX**. It involves three mechanisms:

1. **Shared Memory Segments**
2. **Semaphores**
3. **Message Queues**

More modern programs tend to use **POSIX IPC** methods for all three of these mechanisms, but there are still plenty of **System V IPC** applications found in the wild.

To get an overall summary of **System V IPC** activity on your system, do:

```
$ ipcs
```

```
------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x01114703 0          root       600        1000       6
0x00000000 98305      coop       600        4194304    2          dest
0x00000000 196610     coop       600        4194304    2          dest
0x00000000 23068675   coop       700        1138176    2          dest
0x00000000 23101444   coop       600        393216     2          dest
0x00000000 23134213   coop       600        524288     2          dest
0x00000000 24051718   coop       600        393216     2          dest
0x00000000 23756807   coop       600        524288     2          dest
0x00000000 24018952   coop       600        67108864   2          dest
0x00000000 23363593   coop       700        95408      2          dest
0x00000000 1441811    coop       600        2097152    2          dest

------ Semaphore Arrays --------
key        semid      owner      perms      nsems
0x00000000 98304      apache     600        1
0x00000000 131073     apache     600        1
0x00000000 163842     apache     600        1
```

```
0x00000000 196611     apache     600        1
0x00000000 229380     apache     600        1
```

Note almost all of the currently running shared memory segments have a key of `0` (also known as `IPC_PRIVATE`) which means they are only shared between processes in a parent/child relationship. Furthermore, all but one are marked for destruction when there are no further attachments.

One can gain further information about the processes that have created the segments and last attached to them with:

```
$ ipcs -p
```

```
------ Message Queues PIDs --------
msqid      owner      lspid      lrpid

------ Shared Memory Creator/Last-op PIDs --------
shmid      owner      cpid       lpid
0          root       1023       1023
98305      coop       2265       18780
196610     coop       2138       18775
23068675   coop       989        1663
23101444   coop       989        1663
23134213   coop       989        1663
24051718   coop       20573      1663
23756807   coop       10735      1663
24018952   coop       17875      1663
23363593   coop       989        1663
1441811    coop       2048       20573
```

Thus, by doing:

```
$ ps aux |grep -e 20573 -e 2048
```

```
coop      2048  5.3  3.7 1922996 305660 ?      Rl   Oct27  77:07 /usr/bin/gnome-shell
coop     20573  1.9  1.7 807944 141688 ?       Sl   09:56   0:01 /usr/lib64/thunderbird/thunderbird
coop     20710  0.0  0.0 112652  2312 pts/0    S+   09:57   0:00 grep --color=auto -e 20573 -e 2048
```

we see **thunderbird** is using a shared memory segment created by **gnome-shell**.

Perform these steps on your system and identify the various resources being used and by who. Are there any potential **leaks** (shared resources no longer being used by any active processes) on the system? For example, doing:

```
$ ipcs
```

```
....
------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
....
0x00000000 622601     coop       600        2097152    2          dest
0x0000001a 13303818   coop       666        8196       0
....
```

shows a shared memory segment with no attachments and not marked for destruction. Thus it might persist forever, leaking memory if no subsequent process attaches to it.

Exercise 14.3: Getting Uptime and Load Average

Ascertain how long your system has been up, and also display its load average.

Solution 14.3

A very simple method is just to use the **uptime** utility:

```
$ uptime
```

```
 10:26:40 up  3:19,  5 users,  load average: 1.46, 1.40, 1.19
```

A second method is to look at the first line of output from **top**:

```
$ top | head
```

```
top - 10:28:11 up  3:20,  5 users,  load average: 1.93, 1.52, 1.25
Tasks: 313 total,   1 running, 312 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us,  0.3 sy,  0.0 ni, 98.2 id,  0.5 wa,  0.0 hi,  0.0 si,  0.0
KiB Mem : 16284472 total,  6556792 free,  1029760 used,  8697920 buff/cache
KiB Swap:  8290300 total,  8290300 free,        0 used. 10364220 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2615 coop      20   0  504536 186312  65488 S   6.7  1.1   6:28.30 skype-b+
18248 coop      20   0  655804  50816  30884 S   6.7  0.3   0:20.11 emacs
    1 root      20   0  204912   6508   3956 S   0.0  0.0   0:00.92 systemd
```

A third method is to use **w**:

```
 10:30:51 up  3:23,  5 users,  load average: 0.55, 1.11, 1.14
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
coop     :0       :0               07:08   ?xdm?  16:51   0.19s gdm-session-
coop     pts/0    :0               07:09    2:22m  0.12s  0.12s bash
coop     pts/1    :0               07:09    1:37m  0.42s  0.42s bash
coop     pts/2    :0               07:09    0.00s 51.09s  0.00s w
coop     pts/3    :0               07:09   27:08   0.25s  0.25s bash
```

Exercise 14.4: Background and Foreground Jobs

We are going to launch a graphical program from a terminal window, so that one can no longer type in the window. **gedit** is an easy choice but you can substitute any other program that does this.

1. Open **gedit** on a new file as in

   ```
   $ gedit somefile
   ```

   You can no longer type in the terminal window.

2. While your pointer is over the terminal window, hit CTRL-Z.

   ```
   ^Z
   ```

   ```
   [3]+  Stopped                 gedit somefile
   ```

   You can no longer type in the **gedit** window.

3. With **jobs -l**, see what processes have been launched from this terminal window:

   ```
   $ jobs -l
   ```

   ```
   [1]  17705 Running                 evince *pdf &
   [2]- 18248 Running                 emacs /tmp/hello.tex &
   [3]+ 19827 Stopped                 gedit somefile
   ```

4. Now put the most recent job (**gedit somefile**) in background:

   ```
   $ bg
   ```

```
[3]+ gedit somefile &
```

Now you should be able to type in the **gedit** window.

5. Put the process in foreground again:

```
$ fg
gedit somefile
```

Note you can no longer type in the terminal window.

6. To clean up, suspend the process again and then use **kill** to terminate it:

```
^Z
```

```
[3]+  Stopped                 gedit somefile
```

```
$ jobs -l
```

```
[1]  17705 Running                 evince *pdf &
[2]- 18248 Running                 emacs /tmp/hello.tex &
[3]+ 19827 Stopped                 gedit somefile
```

```
$ kill -9 19827
$ jobs -l
```

```
[1]  17705 Running                 evince *pdf &
[2]- 18248 Running                 emacs /tmp/hello.tex &
[3]+ 19827 Killed                  gedit somefile
```

```
$ jobs -l
```

```
[1]- 17705 Running                 evince *pdf &
[2]- 18248 Running                 emacs /tmp/hello.tex &
```

Exercise 14.5: Using at for Batch Processing in the Future

Schedule a very simple task to run at a future time from now. This can be as simple as running **ls** or **date** and saving the output. (You can use a time as short as one minute in the future.)

Note that the command will run in the directory from which you schedule it with **at**.

Do this:

1. From a short **bash** script.
2. Interactively

Solution 14.5

1. Create the file `testat.sh` containing:

```
#!/bin/bash
date > /tmp/datestamp
```

and then make it executable and queue it up with **at**:

```
$ chmod +x testat.sh
$ at now + 1 minute -f testat.sh
```

You can see if the job is queued up to run with **atq**:

```
$ atq
```

```
17        Wed Apr 22 08:55:00 2015 a student
```

Make sure the job actually ran:

```
$ cat /tmp/datestamp
```

```
Wed Apr 22 08:55:00 CDT 2015
```

What happens if you take the `>/tmp/datestamp` out of the command? (Hint: type **mail** if not prompted to do so!)

2. Interactively it is basically the same procedure. Just queue up the job with:

```
$ at now + 1 minute
at> date > /tmp/datestamp
CTRL-D
$ atq
```

Exercise 14.6: Scheduling a Periodic Task with cron

Set up a **cron** job to do some simple task every day at 10 AM.

Solution 14.6

Create a file named `mycrontab` with the following content:

```
0 10 * * * /tmp/myjob.sh
```

and then create `/tmp/myjob.sh` containing:

SH `/tmp/myjob.sh`

```
#!/bin/bash
echo Hello I am running $0 at $(date)
```

and make it executable:

```
$ chmod +x /tmp/myjob.sh
```

Put it in the **crontab** system with:

```
$ crontab mycrontab
```

and verify it was loaded with:

```
$ crontab -l
```

```
0 10 * * * /tmp/myjob.sh
```

```
$ sudo ls -l /var/spool/cron/student
```

```
-rw------- 1 student student 25 Apr 22 09:59 /var/spool/cron/student
```

```
$ sudo cat /var/spool/cron/student
0 10 * * * /tmp/myjob.sh
```

Note if you don't really want this running every day, printing out messages like:

```
Hello I am running /tmp/myjob.sh at Wed Apr 22 10:03:48 CDT 2015
```

and mailing them to you, you can remove it with:

```
$ crontab -r
```

If the machine is not up at 10 AM on a given day, **anacron** will run the job at a suitable time.

Chapter 15

Process Monitoring

15.1 Labs

Exercise 15.1: Processes

1. Run **ps** with the options `-ef`. Then run it again with the options `aux`. Note the differences in the output.
2. Run **ps** so that only the process ID, priority, nice value, and the process command line are displayed.
3. Start a new **bash** session by typing **bash** at the command line. Start another **bash** session using the **nice** command but this time giving it a nice value of `10`.
4. Run **ps** as in step 2 to note the differences in priority and nice values. Note the process ID of the two **bash** sessions.
5. Change the nice value of one of the **bash** sessions to 15 using **renice**. Once again, observe the change in priority and nice values.
6. Run **top** and watch the output as it changes. Hit `q` to stop the program.

Solution 15.1

1.
```
$ ps -ef
$ ps aux
```

2.
```
$ ps -o pid,pri,ni,cmd
```

```
  PID PRI  NI CMD
 2389  19   0 bash
22079  19   0 ps -o pid,pri,ni,cmd
```

   (Note: There should be no spaces between parameters.)

3.
```
$ bash
$ nice -n 10 bash
$ ps -o pid,pri,ni,cmd
```

```
  2389  19   0 bash
 22115  19   0 bash
 22171   9  10 bash
 22227   9  10 ps -o pid,pri,ni,cmd
```

4. `$ renice 15 -p 22171`
   `$ ps -o pid,pri,ni,cmd`

```
   PID PRI  NI CMD
  2389  19   0 bash
 22115  19   0 bash
 22171   4  15 bash
 22246   4  15 ps -o pid,pri,ni,cmd
```

5. `$ top`

Exercise 15.2: Monitoring Process States

1. Use **dd** to start a background process which reads from `/dev/urandom` and writes to `/dev/null`.
2. Check the process state. What should it be?
3. Bring the process to the foreground using the **fg** command. Then hit `Ctrl-Z`. What does this do? Look at the process state again, what is it?
4. Run the **jobs** program. What does it tell you?
5. Bring the job back to the foreground, then terminate it using **kill** from another window.

Solution 15.2

1. `$ dd if=/dev/urandom of=/dev/null &`

2. `$ ps -C dd -o pid,cmd,stat`

```
 25899 dd if=/dev/urandom of=/dev/ R
```

   Should be `S` or `R`.

3.
```
$ fg
$ ^Z
$ ps -C dd -o pid,cmd,stat
```

```
   PID CMD                       STAT
 25899 dd if=/dev/urandom of=/dev/ T
```

   State should be `T`.

4. Type the **jobs** command. What does it tell you?

```
$ jobs
```

```
 [1]+  Stopped                 dd if=/dev/urandom of=/dev/null
```

5. Bring the job back to the foreground, then kill it using the **kill** command from another window.

```
$ fg
$ kill  25899
```

Chapter 16

Memory Monitoring, Usage and Configuring Swap

16.1 Labs

Exercise 16.1: Managing Swap Space

Examine your current swap space by doing:

```
$ cat /proc/swaps
```

```
Filename                                Type            Size    Used    Priority
/dev/sda11                              partition       4193776 0       -1
```

We will now add more swap space by adding either a new partition or a file. To use a file we can do:

```
$ dd if=/dev/zero of=swpfile bs=1M count=1024
```

```
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 1.30576 s, 822 MB/s
```

```
$ mkswap swpfile
```

```
Setting up swapspace version 1, size = 1048572 KiB
no label, UUID=85bb62e5-84b0-4fdd-848b-4f8a289f0c4c
```

(For a real partition just feed **mkswap** the partition name, but be aware all data on it will be erased!)

Activate the new swap space:

```
$ sudo swapon swpfile
```

```
swapon: /tmp/swpfile: insecure permissions 0664, 0600 suggested.
swapon: /tmp/swpfile: insecure file owner 500, 0 (root) suggested.
```

On Red Hat, Centos, or Fedora

Notice **RHEL** warns us we are being insecure, we really should fix with:

```
$ sudo chown root:root swpfile
$ sudo chmod 600 swpfile
```

We ensure `swpfile` is being used:

```
$ cat /proc/swaps
```

```
Filename                                Type            Size    Used    Priority
/dev/sda11                              partition       4193776 0       -1
/tmp/swpfile                            file            1048572 0       -2
```

Note the `Priority` field; swap partitions or files of lower priority will not be used until higher priority ones are filled.

Remove the swap file from use and delete it to save space:

```
$ sudo swapoff swpfile
$ sudo rm swpfile
```

Exercise 16.2: Invoking the OOM Killer

- When the **Linux** kernel gets under extreme memory pressure it invokes the dreaded **OOM** (**O**ut **O**f **M**emory) **Killer**. This tries to select the “best” process to kill to help the system recover gracefully.
- We are going to force the system to run short on memory and watch what happens. The first thing to do is to open up a terminal window, and in it type:

  ```
  $ sudo tail -f /var/log/messages
  ```

  in order to watch kernel messages as they appear.

> - An even better way to look is furnished by:
>
>   ```
>   $ dmesg -w
>   ```
>
>   as it does not show non-kernel messages.

- This exercise will be easier to perform if we turn off all swap first with the command:

  ```
  $ sudo /sbin/swapoff -a
  ```

  Make sure you turn it back on later with

  ```
  $ sudo /sbin/swapon -a
  ```

- Now we are going to put the system under increasing memory pressure. You are welcome to find your own way of doing it but we also supply a program for consuming the memory:

lab_wastemem.c

```c
/* simple program to defragment memory, J. Cooperstein 2/04
 */

#include <stdio.h>
#include <stdlib.h>
```

```c
#include <unistd.h>
#include <string.h>
#include <sys/sysinfo.h>
#include <signal.h>

#define MB (1024*1024)
#define BS 16                        /* will allocate BS*MB at each step */
#define CHUNK (MB*BS)
#define QUIT_TIME 20
void quit_on_timeout(int sig)
{
        printf("\n\nTime expired, quitting\n");
        exit(EXIT_SUCCESS);
}

int main(int argc, char **argv)
{
        struct sysinfo si;
        int j, m;
        char *c;

        /* get total memory on the system */
        sysinfo(&si);
        m = si.totalram / MB;
        printf("Total System Memory in MB = %d MB\n", m);
        m = (9 * m) / 10;        /* drop 10 percent */
        printf("Using somewhat less: %d MB\n", m);

        if (argc == 2) {
                m = atoi(argv[1]);
                printf("Choosing instead mem = %d MB\n", m);
        }

        signal(SIGALRM, quit_on_timeout);
        printf("Will quite in QUIT_TIME seconds if no normal termination\n");
        alarm(QUIT_TIME);

        for (j = 0; j <= m; j += BS) {
                /* yes we know this is a memory leak, no free,
                 * that's the idea!
                 */
                c = malloc(CHUNK);
                /* just fill the block with j over and over */
                memset(c, j, CHUNK);
                printf("%8d", j);
                fflush(stdout);
        }
        printf("\n\n    Sleeping for 5 seconds\n");
        sleep(5);
        printf("\n\n    Quitting and releasing memory\n");
        exit(EXIT_SUCCESS);
}
```

It takes as an argument how many MB to consume. Keep running it, gradually increasing the amount of memory requested until your system runs out of memory.

Please Note

You should be able to compile the program and run it by just doing:

```
$ gcc -o lab_wastemem lab_wastemem.c
$ ./lab_wastemem 4096
```

which would waste 4 GB. It would be a good idea to run **gnome-system-monitor** or another memory monitoring program while it is running (although the display may freeze for a while!)

- You should see the **OOM** (Out of Memory) killer swoop in and try to kill processes in a struggle to stay alive. Who gets clobbered first?

✓ Solution 16.2

```
Please see SOLUTIONS/s_16/lab_wastemem.c
Please see SOLUTIONS/s_16/lab_waste.sh
```

Chapter 17

I/O Monitoring and Tuning

17.1 Labs

Exercise 17.1: bonnie++

bonnie++ is a widely available benchmarking program that tests and measures the performance of drives and filesystems. It is descended from **bonnie**, an earlier implementation.

Results can be read from the terminal window or directed to a file, and also to a **csv** format (**c**omma **s**eparated **v**alue). Companion programs, **bon_csv2html** and **bon_csv2txt**, can be used to convert to html and plain text output formats.

We recommend you read the **man** page for **bonnie++** before using as it has quite a few options regarding which tests to perform and how exhaustive and stressful they should be. A quick synopsis is obtained with:

```
$ bonnie++ --help
```

```
bonnie++: invalid option -- '-'
usage:
bonnie++ [-d scratch-dir] [-c concurrency] [-s size(MiB)[:chunk-size(b)]]
      [-n number-to-stat[:max-size[:min-size][:num-directories[:chunk-size]]]]
      [-m machine-name] [-r ram-size-in-MiB]
      [-x number-of-tests] [-u uid-to-use:gid-to-use] [-g gid-to-use]
      [-q] [-f] [-b] [-p processes | -y] [-z seed | -Z random-file]
      [-D]

Version: 1.98
```

A quick test can be obtained with a command like:

```
$  time sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

where:

- `-n 0` means don't perform the file creation tests.
- `-u 0` means run as root.
- `-r 100` means pretend you have 100 MB of RAM.

- `-f` means skip per character I/O tests.
- `-b` means do a **fsync** after every write, which forces flushing to disk rather than just writing to cache.
- `-d /mnt` just specifies the directory to place the temporary file created; make sure it has enough space, in this case 300 MB, available.

If you don't supply a figure for your memory size, the program will figure out how much the system has and will create a testing file 2-3 times as large. We are not doing that here because it takes much longer to get a feel for things.

On Red Hat

For **RHEL/CentOS 9**, there is currently no package available but the earlier version works, and can be obtained from the **EPEL 8** repository: https://dl.fedoraproject.org/pub/epel/8/Everything/x86_64/Packages/b/bonnie++-1.98-1.el8.x86_64.rpm and install with:

```
$ sudo dnf localinstall bonnie++-*rpm
```

On an **RHEL/CentOS 8** system:

```
c8:/tmp>sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

```
Using uid:0, gid:0.
Writing intelligently...done
Rewriting...done
Reading intelligently...done
start 'em...done...done...done...done...done...
Version  1.98       ------Sequential Output------ --Sequential Input- --Random-
                    -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Name:Size etc        /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
c8            300M            271m  20  288m  15            +++++ +++  3916  22
Latency                         30us      20us                 12us    18075us

1.98,1.98,c8,1,1616174245,300M,,8192,5,,,277062,20,294543,15,,,+++++,+++,3916,22,,,,,,,,,,,,,,,,,,,\
30us,20us,,12us,18075us,,,,,,
```

On Ubuntu

On an **Ubuntu 20.04** system, running as a virtual machine under the **VMware** hypervisor on the same physical machine:

```
$ sudo bonnie++ -n 0 -u 0 -r 100 -f -b -d /mnt
```

```
sing uid:0, gid:0.
Writing intelligently...done
Rewriting...done
Reading intelligently...done
start 'em...done...done...done...done...done...
Version  1.98       ------Sequential Output------ --Sequential Input- --Random-
                    -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Name:Size etc        /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
ubuntu        300M            287m  88 +++++ +++            +++++ +++ +++++ +++
Latency                       3805us    4650us               2136us    1852us

1.98,1.98,ubuntu,1,1616156268,300M,,8192,5,,,293955,88,+++++,+++,,,+++++,+++,+++++,+++,,,,,,,,,,,,,,,\
,,,,3805us,4650us,,2136us,1852us,,,,,,
```

You can clearly see the drop in performance. Assuming you have saved the previous outputs as a file called bonnie++.out, you can convert the output to html:

```
$ bon_csv2html < bonnie++.out > bonnie++.html
```

or to plain text with:

```
$ bon_csv2txt < bonnie++.out > bonnie++.txt
```

After reading the documentation, try longer and larger, more ambitious tests. Try some of the tests we turned off. If your system is behaving well, save the results for future benchmarking comparisons when the system is sick.

Exercise 17.2: fs_mark

The **fs_mark** benchmark gives a low level bashing to file systems, using heavily asynchronous I/O across multiple directories and drives. It's a rather old program written by Rick Wheeler that has stood the test of time. It can be downloaded from http://sourceforge.net/projects/fsmark/ Once you have obtained the tarball, you can unpack it and compile it with:

```
$ tar zxvf fs_mark-3.3.tgz
$ cd fs_mark
$ make
```

Read the README file as we are only going to touch the surface. If the compile fails with an error like:

```
$ make
```

```
    ....
    /usr/bin/ld: cannot find -lc
```

it is because you haven't installed the **static** version of **glibc**. You can do this on **Red Hat**-based systems by doing:

```
$ sudo dnf install glibc-static
```

and on **SUSE**-related systems with:

```
$ sudo zypper install glibc-devel-static
```

On **Debian**-based systems the relevant static library is installed along with the shared one so no additional package needs to be sought.

On Red Hat, Centos, or Fedora

- On **RHEL 8** glibc-static is available in the codeready-builder repo which may need to be enabled.
- However, on **CentOS 8** it is included in the **PowerTools** repo. If you already have /etc/yum.repos.d/CentOS-PowerTools.repo on your system, then just make sure it has enabled=1 set in the file. Otherwise we have made a copy of this file and placed it in the SOLUTIONS section for this course:

For a test we are going to create 2500 files, each 10 KB in size, and after each write we'll perform an **fsync** to flush out to disk. This can be done in the /tmp directory with the command:

```
$ fs_mark -d /tmp -n 2500 -s 10240
```

While this is running, gather extended **iostat** statistics with:

```
$ iostat -x -d /dev/sda 2 10
```

in another terminal window.

The numbers you should surely note are the number of files per second reported by **fs_mark** and the bandwidth percentage utilized as reported by **iostat**. If this is approaching 100 percent, you are I/O-bound.

Chapter 18

Containers Overview

18.1 Labs

Exercise 18.1: Make a small utility available with docker

Taking our `cowsay` example, build a similar small docker container for the **httpie** utility (https://github.com/httpie/httpie).

Solution 18.1

Alpine Linux solution:

1. **Verify the package name**
   If you just guessed it was “httpie”, that is going to work fine. Also, you can use the package search at: https://pkgs.alpinelinux.org/package/edge/community/x86_64/httpie or check the information in the upstream `README`.
2. **Create a Dockerfile**
   You can use either the `minimal` or `digest` version for this, but in this instance, you probably would use the `minimal` version in practice because the latest version on each build would be desired.
   Additionally, you don't need the extra repo argument for `testing` but it won't hurt anything having it there.

   httpie.Dockerfile

   ```
   FROM alpine:edge
   RUN apk add --no-cache httpie
   CMD ["httpie"]
   ```

3. **Build the image**

   ```
   $ docker build  -f httpie.Dockerfile -t  httpie:latest .
   ```

   ```
   Sending build context to Docker daemon  30.72kB
   Step 1/3 : FROM alpine:edge
    ---> 855f10c27d71
   Step 2/3 : RUN apk add --no-cache httpie
    ---> Running in 2f6b1436e5e3
   fetch https://dl-cdn.alpinelinux.org/alpine/edge/main/x86_64/APKINDEX.tar.gz
   fetch https://dl-cdn.alpinelinux.org/alpine/edge/community/x86_64/APKINDEX.tar.gz
   ```

```
(1/57) Installing libbz2 (1.0.8-r6)
(2/57) Installing libexpat (2.5.0-r2)
(3/57) Installing libffi (3.4.4-r3)
(4/57) Installing gdbm (1.23-r1)
(5/57) Installing xz-libs (5.4.3-r1)
(6/57) Installing libgcc (13.1.1_git20230617-r0)
(7/57) Installing libstdc++ (13.1.1_git20230617-r0)
(8/57) Installing mpdecimal (2.5.1-r2)
(9/57) Installing ncurses-terminfo-base (6.4_p20230625-r0)
(10/57) Installing libncursesw (6.4_p20230625-r0)
(11/57) Installing libpanelw (6.4_p20230625-r0)
(12/57) Installing readline (8.2.1-r2)
(13/57) Installing sqlite-libs (3.42.0-r2)
(14/57) Installing python3 (3.11.4-r0)
(15/57) Installing python3-pycache-pyc0 (3.11.4-r0)
(16/57) Installing pyc (0.1-r0)
(17/57) Installing py3-multidict (6.0.4-r2)
(18/57) Installing py3-multidict-pyc (6.0.4-r2)
(19/57) Installing py3-parsing (3.1.0-r0)
(20/57) Installing py3-parsing-pyc (3.1.0-r0)
(21/57) Installing py3-packaging (23.1-r1)
(22/57) Installing py3-packaging-pyc (23.1-r1)
(23/57) Installing py3-setuptools (68.0.0-r0)
(24/57) Installing py3-setuptools-pyc (68.0.0-r0)
(25/57) Installing py3-pip (23.1.2-r0)
(26/57) Installing py3-pip-pyc (23.1.2-r0)
(27/57) Installing py3-pygments (2.15.1-r0)
(28/57) Installing py3-pygments-pyc (2.15.1-r0)
(29/57) Installing py3-pysocks (1.7.1-r5)
(30/57) Installing py3-pysocks-pyc (1.7.1-r5)
(31/57) Installing py3-certifi (2023.5.7-r0)
(32/57) Installing py3-certifi-pyc (2023.5.7-r0)
(33/57) Installing py3-charset-normalizer (3.1.0-r1)
(34/57) Installing py3-charset-normalizer-pyc (3.1.0-r1)
(35/57) Installing py3-idna (3.4-r4)
(36/57) Installing py3-idna-pyc (3.4-r4)
(37/57) Installing py3-urllib3 (1.26.15-r2)
(38/57) Installing py3-urllib3-pyc (1.26.15-r2)
(39/57) Installing py3-requests (2.31.0-r0)
(40/57) Installing py3-requests-pyc (2.31.0-r0)
(41/57) Installing py3-requests-toolbelt (1.0.0-r0)
(42/57) Installing py3-requests-toolbelt-pyc (1.0.0-r0)
(43/57) Installing py3-attrs (23.1.0-r1)
(44/57) Installing py3-attrs-pyc (23.1.0-r1)
(45/57) Installing py3-mdurl (0.1.2-r2)
(46/57) Installing py3-mdurl-pyc (0.1.2-r2)
(47/57) Installing py3-markdown-it-py (3.0.0-r0)
(48/57) Installing py3-markdown-it-py-pyc (3.0.0-r0)
(49/57) Installing py3-rich (13.4.2-r0)
(50/57) Installing py3-rich-pyc (13.4.2-r0)
(51/57) Installing py3-wheel (0.40.0-r1)
(52/57) Installing py3-wheel-pyc (0.40.0-r1)
(53/57) Installing httpie-pyc (3.2.2-r0)
(54/57) Installing py3-defusedxml-pyc (0.7.1-r4)
(55/57) Installing python3-pyc (3.11.4-r0)
(56/57) Installing py3-defusedxml (0.7.1-r4)
(57/57) Installing httpie (3.2.2-r0)
Executing busybox-1.36.0-r5.trigger
OK: 95 MiB in 72 packages
Removing intermediate container 2f6b1436e5e3
 ---> 227f4a0c96ab
Step 3/3 : ENTRYPOINT ["httpie"]
 ---> Running in f19a66bab6cb
```

```
Removing intermediate container f19a66bab6cb
 ---> c0b613745c84
Successfully built c0b613745c84
Successfully tagged httpie:latest
```

4. **Run the image**

```
$ docker run --rm httpie
```

```
usage: httpie [-h] [--debug] [--traceback] [--version] {cli,plugins} ...
httpie: error: Please specify one of these: 'cli', 'plugins'

This command is only for managing HTTPie plugins.
To send a request, please use the http/https commands:

  $ http POST pie.dev/post hello=world

  $ https POST pie.dev/post hello=world
```

And then try the suggested command:

```
$ docker run --rm httpie https
```

```
usage:
    http [METHOD] URL [REQUEST_ITEM ...]

error:
    the following arguments are required: URL

for more information:
    run 'http --help' or visit https://httpie.io/docs/cli
```

5. *(Optional)* **Add it to your aliases**

```
$ alias https='docker run --rm httpie https -- '
```

```
$ https httpie.io/hello
{"ahoy":["Hello, World!  Thank you for trying out HTTPie ","We hope this will become a
↪   friendship."],"
↪   links":{"homepage":"https://httpie.io","twitter":"https://twitter.com/httpie",
    "discord":"https://httpie.io/discord","github":"https://github.com/httpie"}}
```

Chapter 19

Linux Filesystems and the VFS

19.1 Labs

Exercise 19.1: The tmpfs Special Filesystem

tmpfs is one of many special filesystems used under **Linux**. Some of these are not really used as filesystems, but just take advantage of the filesystem abstraction. However, **tmpfs** is a real filesystem that applications can do I/O on.

Essentially, **tmpfs** functions as a **ramdisk**; it resides purely in memory. But it has some nice properties that old-fashioned conventional ramdisk implementations did not have:

1. The filesystem adjusts its size (and thus the memory that is used) dynamically; it starts at zero and expands as necessary up to the maximum size it was mounted with.

2. If your RAM gets exhausted, **tmpfs** can utilize swap space. (You still can't try to put more in the filesystem than its maximum capacity allows, however.)

3. **tmpfs** does not require having a normal filesystem placed in it, such as **ext3** or **vfat**; it has its own methods for dealing with files and I/O that are aware that it is really just space in memory (it is not actually a block device), and as such are optimized for speed.

   Thus there is no need to pre-format the filesystem with a `mkfs` command; you merely just have to mount it and use it.

Mount a new instance of **tmpfs** anywhere on your directory structure with a command like:

```
$ sudo mkdir /mnt/tmpfs
$ sudo mount -t tmpfs none /mnt/tmpfs
```

See how much space the filesystem has been given and how much it is using:

```
$ df -h /mnt/tmpfs
```

You should see it has been allotted a default value of half of your RAM; however, the usage is zero, and will only start to grow as you place files on `/mnt/tmpfs`.

You could change the allotted size as a mount option as in:

```
$ sudo mount -t tmpfs -o size=1G none /mnt/tmpfs
```

You might try filling it up until you reach full capacity and see what happens. Do not forget to unmount when you are done with:

```
$ sudo umount /mnt/tmpfs
```

Virtually all modern **Linux** distributions mount an instance of **tmpfs** at /dev/shm:

```
$ df -h /dev/shm
```

```
Filesystem     Type   Size  Used Avail Use% Mounted on
tmpfs          tmpfs  3.9G   24M  3.9G   1% /dev/shm
```

Many applications use this such as when they are using **POSIX** shared memory as an inter-process communication mechanism. Any user can create, read and write files in /dev/shm, so it is a good place to create temporary files in memory.

Create some files in /dev/shm and note how the filesystem is filling up with **df**.

In addition, many distributions mount multiple instances of **tmpfs**; for example, on a **RHEL** system:

```
$ df -h | grep ' tmpfs'
```

```
tmpfs          tmpfs     7.8G   38M  7.8G   1% /dev/shm
tmpfs          tmpfs     7.8G   18M  7.8G   1% /run
tmpfs          tmpfs     7.8G     0  7.8G   0% /sys/fs/cgroup
tmpfs          tmpfs     1.6G  1.2M  1.6G   1% /run/user/42
tmpfs          tmpfs     1.6G   56K  1.6G   1% /run/user/1000
```

Notice this was run on a system with 16 GB of ram, so clearly you cannot have all these **tmpfs** filesystems actually using the default ˜8 GB they have each been allotted!

Please Note

Some distributions (such as **Fedora**) may (by default) mount /tmp as a **tmpfs** system; in such cases one has to avoid putting large files in /tmp to avoid running out of memory. Or one can disable this behavior as we discussed earlier when describing /tmp.

Chapter 20

Disk Partitioning

20.1 Labs

Exercise 20.0: Using Loop Back Devices to Emulate Partitions

The exercises in this section will make simple use of **mkfs** for formatting filesystems, and **mount** for mounting them at places in the root filesystem tree. These commands will be explained in detail in the next session.

In the first three exercises in this section, we will use the **loop device** mechanism with or without the **parted** program.

Very Important

For the purposes of later exercises in this course you will need unpartitioned disk space. It need not be large, certainly one or two GB will suffice.

If you are using your own native machine, you either have it or you do not. If you do not, you will have to shrink a partition and the filesystem on it (first!) and then make it available, using **gparted** and/or the steps we have outlined or will outline.

Please Note

If you have real physical unpartitioned disk space you do not **need** to do the following procedures, but it is still a very useful learning exercise.

Exercise 20.1: Using a File as a Disk Partition Image

In this first exercise, we are going to create a file that will be used as a container for a full hard disk partition image, and for all intents and purposes can be used like a real hard partition. In the following exercise, we will show how to put more than one partition on it and have it behave as an entire disk.

1. Create a file full of zeros 1 GB in length:

   ```
   $ dd if=/dev/zero of=imagefile bs=1M count=1024
   ```

   You can make a much smaller file if you like or do not have that much available space in the partition you are creating the file on.

2. Put a filesystem on it:

   ```
   $ mkfs.ext4 imagefile
   ```

   ```
   mke2fs 1.42.9 (28-Dec-2013)
   imagefile is not a block special device.
   Proceed anyway? (y,n) y
   Discarding device blocks: done
   .....
   ```

   Of course you can format with a different filesystem, doing **mkfs.ext3**, **mkfs.vfat**, **mkfs.xfs** etc.

3. Mount it somewhere:

   ```
   $ mkdir mntpoint
   $ sudo mount -o loop imagefile mntpoint
   ```

   You can now use this to your heart's content, putting files etc. on it.

4. When you are done unmount it with:

   ```
   $ sudo umount mntpoint
   ```

An alternative method to using the `loop` option to mount would be:

```
$ sudo losetup /dev/loop2 imagefile
$ sudo mount /dev/loop2 mntpoint
 ....
$ sudo umount mntpoint
$ sudo losetup -d /dev/loop2
```

We will discuss **losetup** in a subsequent exercise, and you can use `/dev/loop[0-7]` but you have to be careful they are not already in use, as we will explain.

You should note that using a loop device file instead of a real partition can be useful, but it is pretty worthless for doing any kind of measurements or benchmarking. This is because you are placing one filesystem layer on top of another, which can only have a negative effect on performance, and mostly you just use the behavior of the underlying filesystem the image file is created on.

Exercise 20.2: Partitioning a Disk Image File

The next level of complication is to divide the container file into multiple partitions, each of which can be used to hold a filesystem, or a swap area.

You can reuse the image file created in the previous exercise or create a new one.

1. Run **fdisk** on `imagefile`:

   ```
   $ sudo fdisk -C 130 imagefile
   ```

   ```
   Device does not contain a recognized partition table
   Building a new DOS disk label with disk identifier 0x6280ced3.
   Welcome to fdisk (util-linux 2.23.2).

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

```
Be careful before using the write command.

Command (m for help):
```

The `-C 130` sets the number of phony cylinders in the drive, and is only necessary in old versions of **fdisk**, which unfortunately you will find on **RHEL 6**. However, it will do no harm on other distributions.

2. Type `m` to get a list of commands:

```
m
```

```
Command (m for help): m
Command action
a   toggle a bootable flag
b   edit bsd disklabel
c   toggle the dos compatibility flag
d   delete a partition
g   create a new empty GPT partition table
G   create an IRIX (SGI) partition table
l   list known partition types
m   print this menu
n   add a new partition
o   create a new empty DOS partition table
p   print the partition table
q   quit without saving changes
s   create a new empty Sun disklabel
t   change a partition's system id
u   change display/entry units
v   verify the partition table
w   write table to disk and exit
x   extra functionality (experts only)

Command (m for help):
```

3. Create a new primary partition and make it 256 MB (or whatever size you would like):

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

```
Partition type:
p   primary (0 primary, 0 extended, 4 free)
e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-2097151, default 2097151): +256M
Partition 1 of type Linux and of size 256 MiB is set
```

4. Add a second primary partition also of 256 MB in size:

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

```
Partition type:
p   primary (1 primary, 0 extended, 3 free)
e   extended
Select (default p): p
Partition number (2-4, default 2): 2
First sector (526336-2097151, default 526336):
Using default value 526336
Last sector, +sectors or +size{K,M,G} (526336-2097151, default 2097151): +256M
Partition 2 of type Linux and of size 256 MiB is set
```

```
Command (m for help): p

Disk imagefile: 1073 MB, 1073741824 bytes, 2097152 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
Disk label type: dos
Disk identifier: 0x6280ced3

Device Boot      StartEnd      Blocks    Id  System
imagefile1          2048       526335      262144   83  Linux
imagefile2          526336      1050623      262144   83  Linux
```

5. Write the partition table to disk and exit:

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

```
The partition table has been altered!

Syncing disks.
```

While this has given us some good practice, we haven't yet seen a way to use the two partitions we just created. We'll start over in the next exercise with a method that lets us do so.

Exercise 20.3: Using losetup and parted

We are going to experiment more with:

- Loop devices and **losetup**
- **parted** to partition at the command line non-interactively.

We expect that you should read the **man pages** for **losetup** and **parted** before doing the following procedures.

Once again, you can reuse the image file or, better still, zero it out and start freshly or with another file.

1. Associate the image file with a **loop** device:

```
$ sudo losetup -f
```

```
/dev/loop1
```

```
$ sudo losetup /dev/loop1 imagefile
```

where the first command finds the first **free** loop device. The reason to do this is you may already be using one or more loop devices. For example, on the system that this is being written on, before the above command is executed:

```
$ losetup -a
```

```
/dev/loop0: []: (/usr/src/KERNELS.sqfs)
```

a **squashfs** compressed, read-only filesystem is already mounted using `/dev/loop0`. (The output of this command will vary with distribution.) If we were to ignore this and use **losetup** on `/dev/loop0` we would almost definitely corrupt the file.

2. Create a disk partition label on the loop device (image file):

```
$ sudo parted -s /dev/loop1 mklabel msdos
```

3. Create three primary partitions on the loop device:

```
$ sudo parted -s /dev/loop1 unit MB mkpart primary ext4 0 256
$ sudo parted -s /dev/loop1 unit MB mkpart primary ext4 256 512
$ sudo parted -s /dev/loop1 unit MB mkpart primary ext4 512 1024
```

4. Check the partition table:

```
$ fdisk -l /dev/loop1
```

```
Disk /dev/loop1: 1073 MB, 1073741824 bytes, 2097152 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
Disk label type: dos
Disk identifier: 0x00050c11

      Device Boot      Start         End      Blocks   Id  System
/dev/loop1p1               1      500000      250000   83  Linux
/dev/loop1p2          500001     1000000      250000   83  Linux
/dev/loop1p3         1000001     2000000      500000   83  Linux
```

5. What happens next depends on what distribution you are on. For example, on **RHEL** and **Ubuntu** you will find new device nodes have been created:

```
$ ls -l /dev/loop1*
```

```
brw-rw---- 1 root disk   7, 1 Oct  7 14:54 /dev/loop1
brw-rw---- 1 root disk 259, 0 Oct  7 14:54 /dev/loop1p1
brw-rw---- 1 root disk 259, 3 Oct  7 14:54 /dev/loop1p2
brw-rw---- 1 root disk 259, 4 Oct  7 14:54 /dev/loop1p3
```

and we will use them in the following.

6. Put filesystems on the partitions:

```
$ sudo mkfs.ext3 /dev/loop1p1
$ sudo mkfs.ext4 /dev/loop1p2
$ sudo mkfs.vfat /dev/loop1p3
```

7. Mount all three filesystems and show they are available:

```
$ mkdir mnt1 mnt2 mnt3

$ sudo mount /dev/loop1p1 mnt1
$ sudo mount /dev/loop1p2 mnt2
$ sudo mount /dev/loop1p3 mnt3

$ df -Th
```

```
Filesystem                Type      Size  Used Avail Use% Mounted on
/dev/sda1                 ext4       29G  8.5G   19G  32% /
....
/dev/loop1p1              ext3      233M  2.1M  219M   1% mnt1
/dev/loop1p2              ext4      233M  2.1M  215M   1% mnt2
/dev/loop1p3              vfat      489M     0  489M   0% mnt3
```

8. After using the filesystems to your heart's content you can unwind it all:

```
$ sudo umount mnt1 mnt2 mnt3
$ rmdir mnt1 mnt2 mnt3
$ sudo losetup -d /dev/loop1
```

Exercise 20.4: Partitioning a Real Hard Disk

If you have real hard disk un-partitioned space available, experiment with **fdisk** to create new partitions, either primary or logical within an extended partition. Write the new partition table to disk and then format and mount the new partitions.

Chapter 21

Filesystem Features: Attributes, Creating, Checking, Usage, Mounting

21.1 Labs

Exercise 21.1: Working with File Attributes

1. With your normal user account use **touch** to create an empty file named `/tmp/appendit`.
2. Use **cat** to append the contents of `/etc/hosts` to `/tmp/appendit`.
3. Compare the contents of `/tmp/appendit` with `/etc/hosts`; there should not be any differences.
4. Try to add the append-only attribute to `/tmp/appendit` by using **chattr**. You should see an error here. Why?
5. As root, retry adding the append-only attribute; this time it should work. Look at the file's extended attributes by using **lsattr**.
6. As a normal user, try and use **cat** to copy over the contents of `/etc/passwd` to `/tmp/appendit`. You should get an error. Why?
7. Try the same thing again as root. You should also get an error. Why?
8. As the normal user, again use the append redirection operator (>>) and try appending the `/etc/passwd` file to `/tmp/appendit`. This should work. Examine the resulting file to confirm.
9. As root, set the immutable attribute on `/tmp/appendit`, and look at the extended attributes again.
10. Try appending output to `/tmp/appendit`, try renaming the file, creating a hard link to the file, and deleting the file as both the normal user and as root.
11. We can remove this file by removing the extended attributes. Do so.

Solution 21.1

1.
```
$ cd /tmp
$ touch appendit
$ ls -l appendit
```

```
-rw-rw-r-- 1 coop coop 0 Oct 23 19:04 appendit
```

2. `$ cat /etc/hosts > appendit`

3. `$ diff /etc/hosts appendit`

4. `$ chattr +a appendit`

```
chattr: Operation not permitted while setting flags on appendit
```

5.
   ```
   $ sudo chattr +a appendit
   $ lsattr appendit
   ```

```
-----a-------e-- appendit
```

6. `$ cat /etc/passwd > appendit`

```
bash: appendit: Operation not permitted
```

7.
   ```
   $ sudo su
   $ cat /etc/passwd > appendit
   ```

```
bash: appendit: Operation not permitted
```

```
$ exit
```

8.
   ```
   $ cat /etc/passwd >> /tmp/appendit
   $ cat appendit
   ```

9.
   ```
   $ sudo chattr +i appendit
   $ lsattr appendit
   ```

```
----ia-------e- appendit
```

10. `$ echo hello >> appendit`

```
-bash: appendit: Permission denied
```

```
$ mv appendit appendit.rename
```

```
mv: cannot move `appendit' to `appendit.rename': Operation not permitted
```

```
$ ln appendit appendit.hardlink
```

```
ln: creating hard link `appendit.hardlink' => `appendit': Operation not permitted
```

```
$ rm -f appendit
```

```
rm: cannot remove `appendit': Operation not permitted
```

```
$ sudo su
$ echo hello >> appendit
```

```
-bash: appendit: Permission denied
```

```
$ mv appendit appendit.rename
```

```
mv: cannot move `appendit' to `appendit.rename': Operation not permitted
```

```
$ ln appendit appendit.hardlink
```

```
ln: creating hard link `appendit.hardlink' => `appendit': Operation not permitted
```

```
$ rm -f appendit
```

```
rm: cannot remove `appendit': Operation not permitted
```

```
$ exit
```

11.
```
$ sudo su
$ lsattr appendit
```

```
----ia-------e- appendit
```

```
$ chattr -ia appendit
$ rm appendit
```

```
rm: remove regular file `appendit'? y
```

```
$ ls appendit
```

```
ls: cannot access appendit: No such file or directory
```

Exercise 21.2: Mounting Options

Fresh Partition or Loopback File

In this exercise you will need to either create a fresh partition, or use a loopback file. The solution will differ slightly and we will provide details of both methods.

1. Use **fdisk** to create a new 250 MB partition on your system, probably on `/dev/sda`. Or create a file full of zeros to use as a loopback file to simulate a new partition.
2. Use **mkfs** to format a new filesystem on the partition or loopback file just created. Do this three times, changing the block size each time. Note the locations of the superblocks, the number of block groups and any other pertinent information, for each case.
3. Create a new subdirectory (say `/mnt/tempdir`) and mount the new filesystem at this location. Verify it has been mounted.
4. Unmount the new filesystem, and then remount it as read-only.
5. Try to create a file in the mounted directory. You should get an error here, why?
6. Unmount the filesystem again.
7. Add a line to your `/etc/fstab` file so that the filesystem will be mounted at boot time.
8. Mount the filesystem.
9. Modify the configuration for the new filesystem so that binary files may not be executed from the filesystem (change defaults to `noexec` in the `/mnt/tempdir` entry). Then remount the filesystem and copy an executable file (such as `/bin/ls`) to `/mnt/tempdir` and try to run it. You should get an error: why?

When you are done you will probably want to clean up by removing the entry from `/etc/fstab`.

Solution 21.2

Physical Partition Solution

1. We won't show the detailed steps in **fdisk**, as it is all ground covered earlier. We will assume the partition created is `/dev/sda11`, just to have something to show.

   ```
   $ sudo fdisk /dev/sda
   ```

   ```
   .....
   w
   $ partprobe -s
   ```

   Sometimes the **partprobe** won't work, and to be sure the system knows about the new partition you have to reboot.

2.
   ```
   $ sudo mkfs -t ext4         -v /dev/sda11
   $ sudo mkfs -t ext4 -b 2048 -v /dev/sda11
   $ sudo mkfs -t ext4 -b 4096 -v /dev/sda11
   ```

   Note the `-v` flag (verbose) will give the requested information; you will see that for a small partition like this the default is 1024 byte blocks.

3.
   ```
   $ sudo mkdir /mnt/tempdir
   $ sudo mount /dev/sda11 /mnt/tempdir
   $ mount | grep tempdir
   ```

4.
   ```
   $ sudo umount /mnt/tempdir
   $ sudo mount -o ro /dev/sda11 /mnt/tempdir
   ```

   If you get an error while unmounting, make sure you are not currently in the directory.

5.
   ```
   $ sudo touch /mnt/tempdir/afile
   ```

6.
   ```
   $ sudo umount /mnt/tempdir
   ```

7. Put this line in `/etc/fstab`:

   ```
   /dev/sda11 /mnt/tempdir ext4 defaults 1 2
   ```

8.
   ```
   $ sudo mount /mnt/tempdir
   $ sudo mount | grep tempdir
   ```

9. Change the line in `/etc/fstab` to:

   ```
   /dev/sda11 /mnt/tempdir ext4 noexec 1 2
   ```

   Then do:

   ```
   $ sudo mount -o remount /mnt/tempdir
   $ sudo cp /bin/ls /mnt/tempdir
   $ /mnt/tempdir/ls
   ```

   You should get an error here, why?

Loopback File Solution

1.
   ```
   $ sudo dd if=/dev/zero of=/imagefile bs=1M count=250
   ```

2.
```
$ sudo mkfs -t ext4         -v
$ sudo mkfs -t ext4 -b 2048 -v /imagefile
$ sudo mkfs -t ext4 -b 4096 -v /imagefile
```

   You will get warned that this is a file and not a partition, just proceed.

   Note the -v flag (verbose) will give the requested information; you will see that for a small partition like this the default is 1024 byte blocks.

3.
```
$ sudo mkdir /mnt/tempdir
$ sudo mount -o loop /imagefile /mnt/tempdir
$ mount | grep tempdir
```

4.
```
$ sudo umount /mnt/tempdir
$ sudo mount -o ro,loop /imagefile /mnt/tempdir
```

   If you get an error while unmounting, make sure you are not currently in the directory.

5.
```
$ sudo touch /mnt/tempdir/afile
```

6.
```
$ sudo umount /mnt/tempdir
```

7. Put this line in /etc/fstab:

   in /etc/fstab

```
/imagefile /mnt/tempdir ext4 loop 1 2
```

8.
```
$ sudo mount /mnt/tempdir
$ sudo mount | grep tempdir
```

9. Change the line in /etc/fstab to:

   in /etc/fstab

```
/imagefile /mnt/tempdir ext4 loop,noexec 1 2
```

   Then do:

```
$ sudo mount -o remount /mnt/tempdir
$ sudo cp /bin/ls /mnt/tempdir
$ /mnt/tempdir/ls
```

   You should get an error here, why?

Exercise 21.3: Configure and test NBD

Overview

In this lab we are going to configure and test **NBD**, communicating via the **loopback** device. Both client and server will be running on the same system.

For the **NBD** export, an empty file will be used.

The /opt directory is to be used for the exported device and configuration files. Production environments may prefer different locations.

CentOS-stream-8 issue

At creation time **CentOS-8-Stream** is not working correctly.

Please use **CentOS-Stream-9** or **Fedora**. Waiting for resolution.

1. First step:

On Red Hat, Centos, or Fedora

Prepare a directory to house the source code downloaded with **git**. Clone, build the **NBD** programs and install.

On Debian, Ubuntu, or Linux Mint

Install the **NBD** client and server packages.

2. Create a `nbd-server.conf` file in the `/opt` directory. A minimal, no comments file is sufficient. (A more interesting configuration file can be found in the `SOLUTIONS` directory.
3. Create two files for exporting and set the ownership to `student`.
4. Start the **nbd-server**.
5. Verify the nbd server responds to a query from the client.
6. Install the **nbd** kernel module, verify the block devices are present.
7. Connect the server supplied image, using the ip address, port to the local block device.
8. Wipe out any existing information on the **NBD**.
9. Using the **NBD**, create a **GPT**.
10. Create a partition on the **NBD**.
11. Add an **ext3** filesystem to the **NBD** and test.

Solution 21.3

1. First step:

On Red Hat, Centos, or Fedora

```
$ mkdir ~/src
$ cd ~/src
$ git clone https://github.com/NetworkBlockDevice/nbd.git
$ cd nbd/
```

The packages `docbook-utils` and `autoconf-archive` may not be installed on your system, install them now:

```
$ sudo dnf install docbook-utils
$ sudo dnf install autoconf-archive
```

Build the programs:

```
$ ./autogen.sh
$ ./configure
$ make
$ sudo make install
```

On Debian, Ubuntu, or Linux Mint

The **Debian**, **Ubuntu**, or **Linux Mint**have deb packages available, install the **NBD** utilities.

```
$ sudo apt install nbd-client nbd-server
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  nbd-client nbd-server
0 upgraded, 2 newly installed, 0 to remove and 43 not upgraded.
\ldots output truncated
```

2. Create the **nbd-server** configuration file in the `/opt` directory. **sudo** will be required to edit files in the `/opt` directory.

   ```
   $ sudo /opt/nbd-server.conf
   ```

/opt/nbd-server.conf

```
[generic]
    allowlist = 1
    listenaddr = 0.0.0.0
    port = 10042
[foo]
    exportname = /opt/dsk1
    readonly = false
[bar]
    exportname = /opt/dsk2
```

3.
   ```
   sudo dd if=/dev/zero of=/opt/dsk1 status=progress bs=100M count=5
   sudo dd if=/dev/zero of=/opt/dsk2 status=progress bs=100M count=5
   sudo chmod 777 /opt
   sudo chown student.student /opt/*
   ```

4.
   ```
   $ sudo nbd-server -C /opt/nbd-server.conf
   ```

5.
   ```
   $ sudo nbd-client -l 127.0.0.1 -p 10042
   ```

   ```
   Negotiation: ..
   foo
   bar
   ```

6.
   ```
   $ sudo modprobe -i nbd
   $ ls /dev/nbd*
   ```

   ```
   /dev/nbd0   /dev/nbd11  /dev/nbd14  /dev/nbd3  /dev/nbd6  /dev/nbd9
   /dev/nbd1   /dev/nbd12  /dev/nbd15  /dev/nbd4  /dev/nbd7
   /dev/nbd10  /dev/nbd13  /dev/nbd2   /dev/nbd5  /dev/nbd8
   ```

7.
   ```
   $ sudo nbd-client -N foo  127.0.0.1 10042 /dev/nbd0
   ```

8. Wipe out the NBD:

   ```
   $ sudo dd if=/dev/zero of=/dev/nbd0 bs=1M count=5
   ```

```
5+0 records in
5+0 records out
5242880 bytes (5.2 MB, 5.0 MiB) copied, 0.00161564 s, 3.2 GB/s
```

9. Create a GPT label on the NBD:

```
$ sudo su -c "echo 'label: gpt' | sfdisk /dev/nbd0"
```

```
Checking that no-one is using this disk right now ... OK

Disk /dev/nbd0: 4 GiB, 4294967296 bytes, 8388608 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

>>> Script header accepted.
>>> Done.
Created a new GPT disklabel (GUID: 000C20C1-6929-424F-93E1-28319DE1FF1F).

New situation:
Disklabel type: gpt
Disk identifier: 000C20C1-6929-424F-93E1-28319DE1FF1F

The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

10. Add a partition to the NBD:

```
$ sudo su -c "echo ';' | sfdisk /dev/nbd0"
```

```
Checking that no-one is using this disk right now ... OK

Disk /dev/nbd0: 4 GiB, 4294967296 bytes, 8388608 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: 000C20C1-6929-424F-93E1-28319DE1FF1F

Old situation:

>>> Created a new GPT disklabel (GUID: EFFC087B-A15C-7D42-809C-4BDED58EE238).
/dev/nbd0p1: Created a new partition 1 of type 'Linux filesystem' and of size 4 GiB.
/dev/nbd0p2: Done.

New situation:
Disklabel type: gpt
Disk identifier: EFFC087B-A15C-7D42-809C-4BDED58EE238

Device       Start     End Sectors Size Type
/dev/nbd0p1   2048 8388574 8386527   4G Linux filesystem

The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

11. Add an **ext3** filesystem to the **NBD** and test:

```
$ sudo mkfs.ext3 -L nbd-foo /dev/nbd0
```

```
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 128000 4k blocks and 128000 inodes
Filesystem UUID: 6a01ff5a-9ad5-4f43-8050-9f2b8df26c14
Superblock backups stored on blocks:
        32768, 98304

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
$ sudo mount LABEL=nbd-foo /mnt
$ sudo touch /mnt/file1
$ ls -l /mnt
```

```
total 16
-rw-r--r-- 1 root root     0 Jun  9 10:58 file1
drwx------ 2 root root 16384 Jun  9 10:56 lost+found
```

Chapter 22

The Ext4 Filesystems

22.1 Labs

Exercise 22.1: Defragmentation

Newcomers to **Linux** are often surprised at the lack of mention of filesystem **defragmentation** tools, since such programs are routinely used in the **Windows** world.

However, native filesystems in **UNIX**-type operating systems, including **Linux**, tend not to suffer serious problems with filesystem fragmentation.

This is primarily because they do not try to cram files onto the innermost disk regions where access times are faster. Instead, they spread free space out throughout the disk, so that when a file has to be created there is a much better chance that a region of free blocks big enough can be found to contain the entire file in either just one or a small number of pieces.

For modern hardware, the concept of innermost disk regions is obscured by the hardware anyway.

Don't do this

For **SSDs** defragmentation can actually shorten the lifespan of the storage media due to finite read/erase/write cycles. On smart operating systems defragmentation is turned off by default on **SSD** drives.

Furthermore, the newer **journalling** filesystems (including **ext4**) work with **extents** (large contiguous regions) by design.

However, there does exist a tool for de-fragmenting **ext4** filesystems:

```
$ sudo e4defrag
```

```
Usage   : e4defrag [-v] file...| directory...| device...
        : e4defrag  -c  file...| directory...| device...
```

e4defrag is part of the **e2fsprogs** package and should be on all modern **Linux** distributions.

The only two options are:

- `-v`: Be verbose.

- `-c`: Don't actually do anything, just analyze and report.

The argument can be:

- A file
- A directory
- An entire device

Examples:

```
$ sudo e4defrag -c /var/log
```

```
<Fragmented files>                              now/best       size/ext
1. /var/log/lastlog                               5/1              9 KB
2. /var/log/sa/sa24                               3/1             80 KB
3. /var/log/rhsm/rhsm.log                         2/1            142 KB
4. /var/log/messages                              2/1           4590 KB
5. /var/log/Xorg.1.log.old                        1/1             36 KB

 Total/best extents                             120/112
 Average size per extent                        220 KB
 Fragmentation score                            1
 [0-30 no problem: 31-55 a little bit fragmented: 56- needs defrag]
 This directory (/var/log) does not need defragmentation.
 Done.
```

```
$ sudo e4defrag  /var/log
```

```
ext4 defragmentation for directory(/var/log)
[2/152]/var/log/Xorg.2.log:     100%    [ OK ]
[3/152]/var/log/Xorg.0.log.old: 100%    [ OK ]
[4/152]/var/log/messages-20141019.gz:   100%    [ OK ]
[5/152]/var/log/boot.log:       100%    [ OK ]
[7/152]/var/log/cups/page_log-20140924.gz:      100%    [ OK ]
[8/152]/var/log/cups/access_log-20141019.gz:    100%    [ OK ]
[9/152]/var/log/cups/access_log:        100%    [ OK ]
[10/152]/var/log/cups/error_log-20141018.gz:    100%    [ OK ]
[11/152]/var/log/cups/error_log-20141019.gz:    100%    [ OK ]
[12/152]/var/log/cups/access_log-20141018.gz:   100%    [ OK ]
[14/152]/var/log/cups/page_log-20141018.gz:     100%    [ OK ]
...
[152/152]/var/log/Xorg.1.log.old:       100%    [ OK ]

        Success:                        [ 112/152 ]
        Failure:                        [ 40/152 ]
```

Try running **e4defrag** on various files, directories, and entire devices, always trying with `-c` first.

You will generally find that **Linux** filesystems only tend to need defragmentation when they get very full, over 90 percent or so, or when they are small and have relatively large files, like when a **boot** partition is used.

Exercise 22.2: Modifying Filesystem Parameters with tune2fs

We are going to fiddle with some properties of a formatted **ext4** filesystem. This does not require unmounting the filesystem first.

In the below you can work with an image file you create as in:

```
$ dd if=/dev/zero of=imagefile bs=1M count=1024
$ mkfs.ext4 imagefile
```

or you can substitute `/dev/sdaX` (using whatever partition the filesystem you want to modify is mounted on) for `imagefile`.

1. Using **dumpe2fs**, obtain information about the filesystem whose properties you want to adjust.
   Display:
   - The maximum mount count setting (after which a filesystem check will be forced.)
   - The `Check interval` (the amount of time after which a filesystem check is forced)
   - The number of blocks reserved, and the total number of blocks
2. Change:
   - The maximum mount count to `30`.
   - The `Check interval` to three weeks.
   - The percentage of blocks reserved to 10 percent.
3. Using **dumpe2fs**, once again obtain information about the filesystem and compare with the original output.

Solution 22.2

1. `$ dumpe2fs imagefile > dumpe2fs-output-initial`

```
dumpe2fs 1.42.9 (28-Dec-2013)
```

`$ grep -i  -e "Mount count" -e "Check interval" -e "Block Count" dumpe2fs-output-initial`

```
Block count:              262144
Reserved block count:     13107
Mount count:              0
Maximum mount count:      -1
Check interval:           0 (<none>)
```

2. `$ tune2fs -c 30 imagefile`

```
tune2fs 1.42.9 (28-Dec-2013)
Setting maximal mount count to 30
```

`$ tune2fs -i 3w imagefile`

```
tune2fs 1.42.9 (28-Dec-2013)
Setting interval between checks to 1814400 seconds
```

`$ tune2fs -m 10 imagefile`

```
tune2fs 1.42.9 (28-Dec-2013)
Setting reserved blocks percentage to 10% (26214 blocks)
```

3. `$ dumpe2fs imagefile > dumpe2fs-output-final`

```
dumpe2fs 1.42.9 (28-Dec-2013)
```

`$ grep -i  -e "Mount count" -e "Check interval" -e "Block Count" dumpe2fs-output-final`

```
Block count:              262144
Reserved block count:     26214
Mount count:              0
Maximum mount count:      30
```

```
Check interval:            1814400 (3 weeks)
```

```
$ diff dumpe2fs-output-initial dumpe2fs-output-final
```

```
14c14
< Reserved block count:     13107
---
> Reserved block count:     26214
29c29
< Last write time:          Wed Oct 26 14:26:19 2016
---
> Last write time:          Wed Oct 26 14:26:20 2016
31c31
< Maximum mount count:      -1
---
> Maximum mount count:      30
33c33,34
< Check interval:           0 (<none>)
---
> Check interval:           1814400 (3 weeks)
> Next check after:         Wed Nov 16 13:26:16 2016
```

Chapter 23

Logical Volume Management (LVM)

23.1 Labs

Exercise 23.1: Logical Volumes

We are going to create a logical volume using two 250 MB partitions. We are going to assume you have real partitionable disk space available.

1. Create two 250 MB partitions of type logical volume (`8e`).
2. Convert the partitions to physical volumes.
3. Create a volume group named `myvg` and add the two physical volumes to it. Use the default extent size.
4. Allocate a 300 MB logical volume named `mylvm` from volume group `myvg`.
5. Format and mount the logical volume `mylvm` at `/mylvm`
6. Use **lvdisplay** to view information about the logical volume.
7. Grow the logical volume and corresponding filesystem to 350 MB.

Solution 23.1

1. Execute:

   ```
   $ sudo fdisk /dev/sda
   ```

   using whatever hard disk is appropriate, and create the two partitions. While in **fdisk**, typing `t` will let you set the partition type to `8e`. While it doesn't matter if you don't set the type, it is a good idea to lessen confusion. Use `w` to rewrite the partition table and exit, and then

   ```
   $ sudo partprobe -s
   ```

   or reboot to make sure the new partitions take effect.

2. Assuming the new partitions are `/dev/sdaX` and `/dev/sdaY`:

   ```
   $ sudo pvcreate /dev/sdaX
   $ sudo pvcreate /dev/sdaY
   $ sudo pvdisplay
   ```

3.
```
$ sudo vgcreate myvg /dev/sdaX /dev/sdaY
$ sudo vgdisplay
```

4.
```
$ sudo lvcreate -L 300M -n mylvm myvg
$ sudo lvdisplay
```

5.
```
$ sudo mkfs.ext4 /dev/myvg/mylvm
$ sudo mkdir /mylvm
$ sudo mount /dev/myvg/mylvm /mylvm
```

   If you want the mount to be persistent, edit `/etc/fstab` to include the line:

   in `/etc/fstab`

```
/dev/myvg/mylvm /mylvm ext4 defaults 0 0
```

6.
```
$ sudo lvdisplay
```

7.
```
$ df -h
$ sudo lvresize -r -L 350M /dev/myvg/mylvm
$ df -h
```

   or

```
$ sudo lvresize -r -L +50M /dev/myvg/mylvm
```

   or with older methods you can do:

```
$ df -h
$ sudo lvextend -L 350M /dev/myvg/mylvm
$ sudo resize2fs /dev/myvg/mylvm
$ df -h
```

Chapter 24

Kernel Services and Configuration

24.1 Labs

Exercise 24.1: System Tunables with sysctl

1. Check if you can **ping** your own system.
2. Check the current value of `net.ipv4.icmp_echo_ignore_all`, which is used to turn on and off whether your system will respond to **ping**. A value of `0` allows your system to respond to pings.
3. Set the value to `1` using the **sysctl** command line utility and then check if pings are responded to.
4. Set the value back to `0` and show the original behavior in restored.
5. Now change the value by modifying `/etc/sysctl.conf` and force the system to activate this setting file without a reboot.
6. Check that this worked properly.

You will probably want to reset your system to have its original behavior when you are done.

Solution 24.1

You can use either `localhost`, `127.0.0.1` (loopback address) or your actual IP address for target of **ping** below.

1.
```
$ ping localhost
```
2.
```
$ sysctl net.ipv4.icmp_echo_ignore_all
```
3.
```
$ sudo sysctl net.ipv4.icmp_echo_ignore_all=1
$ ping localhost
```
4.
```
$ sudo sysctl net.ipv4.icmp_echo_ignore_all=0
$ ping localhost
```
5. Add the following line to `/etc/sysctl.conf`:

in /etc/sysctl.conf

```
net.ipv4.icmp_echo_ignore_all=1
```

and then do:

```
$ sysctl -p
```

6.
```
$ sysctl net.ipv4.icmp_echo_ignore_all
$ ping localhost
```

Since the changes to `/etc/sysctl.conf` are persistent, you probably want to restore things to its previous state.

Exercise 24.2: Changing the Maximum Process ID

The normal behavior of a **Linux** system is that process IDs start out at `PID=1` for the **init** process, the first user process on the system, and then go up sequentially as new processes are constantly created (and die as well.)

However, when the PID reaches the value shown in /proc/sys/kernel/pid_max, which is conventionally 32768 (32K), they will wrap around to lower numbers. If nothing else, this means you can't have more than 32K processes on the system since there are only that many slots for PIDs.

1. Obtain the current maximum PID value.
2. Find out what current PIDs are being issued
3. Reset `pid_max` to a lower value than the ones currently being issued.
4. Start a new process and see what it gets as a PID.

Solution 24.2

Very Important

In the below we are going to use two methods, one involving **sysctl**, the other directly echoing values to /proc/sys/kernel/pid_max. Note that the **echo** method requires you to be root; **sudo** won't work. We'll leave it to you to figure out why, if you don't already know!

1.
```
$ sysctl kernel.pid_max
$ cat /proc/sys/kernel/pid_max
```

2. Type:
```
$ cat &
```
```
[1] 29222
```
```
$ kill -9 29222
```

3.
```
$ sudo sysctl kernel.pid_max=24000
$ echo 24000 >  /proc/sys/kernel/pid_max # This must be done as root
$ cat /proc/sys/kernel/pid_max
```

4.
```
$ cat &
```
```
[2] 311
```
```
$ kill -9 311
```

Please Note

Note that when starting over, the kernel begins at PID=300, not a lower value. You might notice that assigning PIDs to new processes is actually not trivial; since the system may have already turned over, the kernel always has to check when generating new PIDs that the PID is not already in use. The **Linux** kernel has a very efficient way of doing this that does not depend on the number of processes on the system.

Chapter 25

Kernel Modules

25.1 Labs

Exercise 25.1: Kernel Modules

1. List all currently loaded kernel modules on your system.
2. Load a currently unloaded module on your system.

   If you are running a distribution kernel, this is easy to find; you can simply look in the /lib/modules/⟨kernel-version⟩/kernel/drivers/net directory and grab one. (Distribution kernels come with drivers for every device, filesystem, network protocol etc. that a system might need.) However, if you are running a custom kernel you may not have many unloaded modules compiled.

   A choice that will usually work is to pick either `e1000.ko` or `e1000e.ko`, as while these gigabit Ethernet drivers are quite common, it is very unlikely both would be loaded at once.
3. Re-list all loaded kernel modules and see if your module was indeed loaded.
4. Remove the loaded module from your system.
5. Re-list again and see if your module was properly removed.

Solution 25.1

1. `$ lsmod`
2. In the following, substitute whatever module name you used for `e1000e`. Either of these methods work but, of course, the second is easier.

   ```
   $ sudo insmod /lib/modules/$(uname -r)/kernel/drivers/net/ethernet/intel/e1000e.ko
   $ sudo /sbin/modprobe e1000e
   ```
3. `$ lsmod | grep e1000e`
4. Once again, either method works.

   ```
   $ sudo rmmod e1000e
   $ sudo modprobe -r e1000e
   ```
5. `$ lsmod | grep e1000e`

Chapter 26

Devices and udev

26.1 Labs

Exercise 26.1: `udev`

1. Create and implement a rule on your system that will create a symlink called `myusb` when a **USB** device is plugged in.
2. Plug in a **USB** device to your system. It can be a pen drive, mouse, webcam, etc.

   Note: If you are running a virtual machine under a hypervisor, you will have to make sure the **USB** device is seen by the guest, which usually is just a mouse click which also disconnects it from the host.
3. Get a listing of the `/dev` directory and see if your symlink was created.
4. Remove the **USB** device. (If it is a drive you should always **umount** it first for safety.)
5. See if your symbolic link still exists in `/dev`.

Solution 26.1

1. Create a file named `/etc/udev/rules.d/75-myusb.rules` and have it include just one line of content:

   ```
   $ cat /etc/udev/rules.d/75-myusb.rules
   ```

   ```
   SUBSYSTEM=="usb", SYMLINK+="myusb"
   ```

   Do not use the deprecated key value `BUS` in place of `SUBSYSTEM`, as recent versions of **udev** have removed it.

   Note the name of this file really does not matter. If there was an `ACTION` component to the rule the system would execute it; look at other rules for examples.
2. Plug in a device.

3.
   ```
   $ ls -lF /dev | grep myusb
   ```
4. If the device has been mounted:

   ```
   $ umount /media/whatever
   ```

   where `/media/whatever` is the mount point. Safely remove the device.

5.
   ```
   $ ls -lF /dev | grep myusb
   ```

Chapter 27

Network Addresses

27.1 Labs

Exercise 27.1: Configure and enable a stratum 3 NTP server. Connect your server to the NTP pool as a client

Solution 27.1

1. Ensure the NTP daemon is installed using your favorite installer:

```
# yum install ntp
# zypper install ntp
# apt install ntp
```

2. Configure the NTP server to query the NTP pool and allow for anonymous traffic.

   Edit `/etc/ntp.conf` and change the `server X.X.X.X` lines to match these:

/etc/ntp.conf

```
server 0.pool.ntp.org
server 1.pool.ntp.org
server 2.pool.ntp.org
server 3.pool.ntp.org
```

3. Restart **ntp** server with the appropriate command:

```
# systemctl restart ntpd
# systemctl restart ntp
```

4. Query your new timeserver:

```
$ ntpq -p
```

Please Note

You may have to wait a few minutes for the timeservers to sync prior to getting any output.

Chapter 28

Network Devices and Configuration

28.1 Labs

Exercise 28.1: Adding a Static Hostname

In this exercise we will add entries to the local host database.

1. Open /etc/hosts and add an entry for mysystem.mydomain that will point to the IP address associated with your network card.
2. Add a second entry that will make all references to ad.doubleclick.net point to 127.0.0.1.
3. As an optional exercise, download the host file from: http://winhelp2002.mvps.org/hosts2.htm or more directly from http://winhelp2002.mvps.org/hosts.txt, and install it on your system. Do you notice any difference using your browser with and without the new host file in place?

Solution 28.1

1.
```
$ sudo sh -c "echo 192.168.1.180    mysystem.mydomain >> /etc/hosts"
$ ping mysystem.mydomain
```

2.
```
$ sudo sh -c "echo 127.0.0.1       ad.doubleclick.net >> /etc/hosts"
$ ping ad.doubleclick.net
```

3.
```
$ wget http://winhelp2002.mvps.org/hosts.txt
```

```
--2014-11-01 08:57:12--  http://winhelp2002.mvps.org/hosts.txt
Resolving winhelp2002.mvps.org (winhelp2002.mvps.org)... 216.155.126.40
Connecting to winhelp2002.mvps.org (winhelp2002.mvps.org)|216.155.126.40|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 514744 (503K) [text/plain]
Saving to: hosts.txt

100%[======================================>] 514,744      977KB/s   in 0.5s

2014-11-01 08:57:13 (977 KB/s) - hosts.txt saved [514744/514744]
```

```
$ sudo sh -c "cat hosts.txt >> /etc/hosts"
```

Exercise 28.2: Verify DNS information with dig

In this exercise we will verify that a forward `IPv4 A` record matches its reverse `PTR` record for a mail server with **dig**.

1. Make sure you have bind installed. If not:

On Debian, Ubuntu, or Linux Mint

apt install bind9-dnsutils

On Red Hat, Centos, or Fedora

dnf install bind-utils

2. Find the mail server IP address for **protonmail.com**.
3. Verify that the reverse lookup for the IP address matches.

Solution 28.2

1.
```
$ dig +short -t mx protonmail.com
```

```
10 mailsec.protonmail.ch.
5 mail.protonmail.ch.
```

```
$ dig +short -t A mail.protonmail.ch
```

```
185.70.42.128
185.205.70.128
176.119.200.128
```

2.
```
$ dig +short -x 185.205.70.128
```

```
mail.protonmail.ch.
```

Exercise 28.3: Adding a Network Interface Alias/Address using nmcli

We are going to add an additional IPv4 address to your system and make it persistent. We will do this without editing files under `/dev` directly, using **nmcli**.

1. First obtain your current internet address and interface name:

```
$ sudo nmcli con
```

```
NAME          UUID                                  TYPE           DEVICE
Auto Ethernet 1c46bf37-2e4c-460d-8b20-421540f7d0e2  802-3-ethernet ens33
virbr0        a84a332f-38e3-445a-a377-4363a8eb963f  bridge         virbr0
```

shows the name of the connection is `Auto Ethernet`.

```
$ sudo nmcli con show "Auto Ethernet" | grep IP4.ADDRESS
```

```
IP4.ADDRESS[1]:                         172.16.2.135/24
```

shows the address as `172.16.2.135` Note that this command shows all information about the connection and you could have specified the `UUID` instead of the `NAME` as in:

```
$ nmcli con show 1c46bf37-2e4c-460d-8b20-421540f7d0e2
```

2. Add a new address that your machine can be seen by:

```
$ sudo nmcli con modify "Auto Ethernet" +ipv4.addresses 172.16.2.140/24
```

3. Activate it and test to see if it is there:

```
$ sudo nmcli con up "Auto Ethernet"
```

```
Connection successfully activated (D-Bus active path:
↪  /org/freedesktop/NetworkManager/ActiveConnection/6)

$ ping -c 3 172.16.2.140
PING 172.16.2.140 (172.16.2.140) 56(84) bytes of data.
64 bytes from 172.16.2.140: icmp_seq=1 ttl=64 time=0.038 ms
64 bytes from 172.16.2.140: icmp_seq=2 ttl=64 time=0.034 ms
64 bytes from 172.16.2.140: icmp_seq=3 ttl=64 time=0.032 ms

--- 172.16.2.140 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.032/0.034/0.038/0.007 ms
```

4. Clean up by removing the alias:

```
$ sudo nmcli con modify  "Auto Ethernet" -ipv4.addresses 172.16.2.140/24
...
$ sudo nmcli con up "Auto Ethernet"
...
```

Exercise 28.4: Adding a Static Route using nmcli

We are going to add a static IPv4 route address to your system and make it persistent. We will do this without editing files under `/dev` directly, using **nmcli**.

1. Begin by examining your current routing tables, using **ip route**:

```
$ ip route
```

```
default via 172.16.2.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
172.16.2.0/24 dev ens33  proto kernel  scope link  src 172.16.2.135  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown
```

2. Add a new route using **nmcli**:

```
$ sudo nmcli conn mod "Auto Ethernet" +ipv4.routes "192.168.100.0/24 172.16.2.1"
```

3. Note it has not yet taken effect:

```
$ ip route
```

```
default via 172.16.2.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
172.16.2.0/24 dev ens33  proto kernel  scope link  src 172.16.2.135  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown
```

4. Reload the interface to have it take effect and show it has:

```
$ sudo nmcli conn up "Auto Ethernet"
```

```
Connection successfully activated (D-Bus active path:
↪  /org/freedesktop/NetworkManager/ActiveConnection/25)
```

```
$ ip route
```

```
default via 172.16.2.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
172.16.2.0/24 dev ens33  proto kernel  scope link  src 172.16.2.135  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown
192.168.100.0/24 via 172.16.2.1 dev ens33
```

5. Reboot and verify the route has taken effect (i.e., it is **persistent**: If so remove it:

```
$ ip route
```

```
default via 172.16.2.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
172.16.2.0/24 dev ens33  proto kernel  scope link  src 172.16.2.135  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown
192.168.100.0/24 via 172.16.2.1 dev ens33
```

```
$ sudo nmcli conn mod "Auto Ethernet" -ipv4.routes "192.168.100.0/24 172.16.2.1"

$ sudo nmcli conn up "Auto Ethernet"
```

```
Connection successfully activated (D-Bus active path:
↪  /org/freedesktop/NetworkManager/ActiveConnection/3)
$ ip route
default via 172.16.2.2 dev ens33  proto static  metric 100
169.254.0.0/16 dev ens33  scope link  metric 1000
172.16.2.0/24 dev ens33  proto kernel  scope link  src 172.16.2.135  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown
```

6. Note you can set a route with **ip** from the command line but it won't survive a reboot as in:

```
$ sudo ip route add 192.168.100.0/24 via 172.16.2.1
...
```

You can verify that a route established this way is not persistent.

Chapter 29

LDAP **

29.1 Labs **

Exercise 29.1: Centralized Authentication using LDAP and TLS - Install an LDAP server appliance

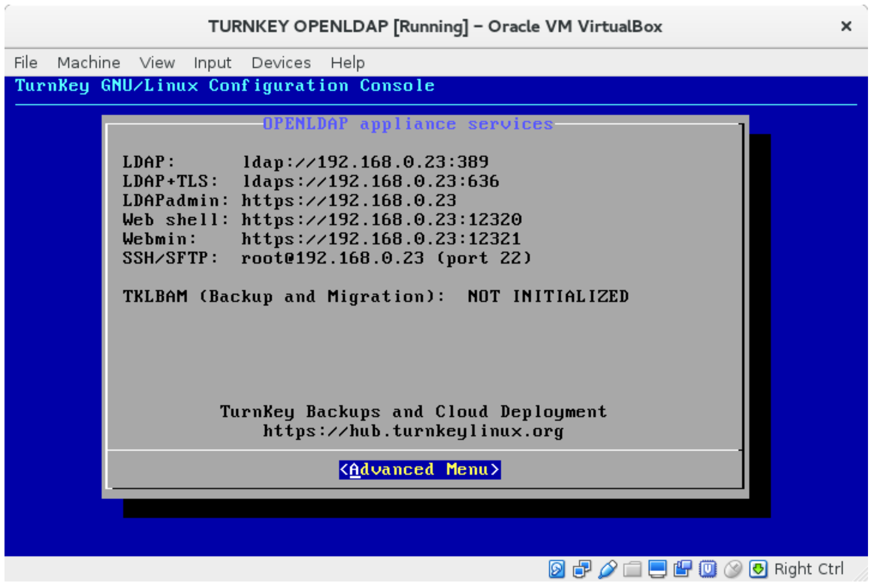
Resource Requirements

This exercise requires an additional Virtual Machine be installed. It is a small **LDAP server** that is used for testing our client configurations. The **LDAP** appliance uses about 512MB of memory and 2GB of disk space. Please complete this exercise outside of class.

This exercise will install a **LDAP** server appliance for centralized network authentication. The **turnkey openldap** appliance server was chosen for its availability on different distributions. Installing this appliance will create a new **VM.** The **appliance VM** is quite small, assigning only 512MiB of memory and using less than 1GB of disk space. The disk space is dynamically grow-able to 20GB but our usage remains quite low. There are many options to obtain the appliance and one option, the OVA image that is compatible with **VMWare** and **Virtual-Box** is available in the following directory. Another option is a **vmdk** image file that works with **VMWare Player** and **VMWare Workstation** is also available in the directory below.

https://training.linuxfoundation.org/cm/LFS307

Please see https://www.turnkeylinux.org/openldap for available options.


TURNKEY OPENLDAP [Running] – Oracle VM VirtualBox
File Machine View Input Devices Help
TurnKey GNU/Linux Configuration Console
OPENLDAP appliance services
<Advanced Menu>
Right Ctrl

```
LDAP:       ldap://192.168.0.23:389
LDAP+TLS:   ldaps://192.168.0.23:636
LDAPadmin:  https://192.168.0.23
Web shell:  https://192.168.0.23:12320
Webmin:     https://192.168.0.23:12321
SSH/SFTP:   root@192.168.0.23 (port 22)

TKLBAM (Backup and Migration):  NOT INITIALIZED

         TurnKey Backups and Cloud Deployment
             https://hub.turnkeylinux.org
```

Figure 29.1: **Turnkey OpenLDAP Appliance Services**

1. Import the appliance
   - Verify the file `turnkey-openldap-14.1-jessie-amd64.ova` is available (version may be different).
   - Open the **Virtual-Box Manager**.
   - Select `File -> Import Appliance`
   - Enter location (or browse) of the **OpenLADP appliance**.
   - Select **Next** and continue as prompted.
2. Start the **TURNKEY OPENLDAP** virtual machine just created.
3. Configure the LDAP appliance on first boot with the following parameters:
   - `Root's password is  "pa$$w0rd"`
   - `Admin's password id "pa$$w0rd"`
   - `The domain is "example.com"`
   - `No "HUB","System Notification" or "Update" services`

   The **Turnkey OpenLDAP Appliance Summary** should be visible, this is the normal **running** display of the appliance.

Exercise 29.2: Centralized Authentication using LDAP and TLS - testing the LDAP server and adding users.

In this exercise we will verify the basic operation of the **LDAP** server. Specifically the ability of the server to respond to queries, and add a **POSIX** group and user. Once the user is added we can monitor the connection with **wireshark** and observe the plain text transfer of information to and from the **LDAP** server. Since it is a poor security policy to have clear text user information, we will encrypt the data stream.

Very Important

This lab is going to use clear text when communication with the **ldap** server. The use of encryption keys is out of scope for this course. Clear text is **NOT RECOMMENDED** in any sort of production environment. To accommodate our lab environment we will disable certificate verification. The ability to not verify the keys is handy in our lab but **DO NOT** use this technique on any production systems.

This exercise will be using the machine used in previous exercises to communicate with the **LDAP** server.

It is assumed that the `ready-for.sh` script has been run, the solutions and resource files loaded and extracted to a directory.

1. Verify or install openldap-clients packages, use the appropriate command.
   Ubuntu

   ```
   # apt install sssd sssd-ldap sssd-tools oddjob-mkhomedir ldap-utils
   ```

   CentOS

   ```
   # yum install sssd sssd-ldap sssd-tools oddjob-mkhomedir openldap-clients
   ```

2. Using the **ip address** of the **LDAP** server displayed in the **Turnkey OpenLDAP Appliance Summary** screen in the previous exercise, perform a simple **LDAP** search of the **example.com** domain. We should see information about the base records for the domain example.com

   ```
   # ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub"objectclass=*"
   ```

   ```
   # extended LDIF
   #
   # LDAPv3
   # base <dc=example,dc=com> with scope subtree
   # filter: (objectclass=*)
   # requesting: ALL
   #

   # example.com
   dn: dc=example,dc=com
   objectClass: top
   objectClass: dcObject
   objectClass: organization
   o: example.com
   dc: example

   # admin, example.com
   dn: cn=admin,dc=example,dc=com
   objectClass: simpleSecurityObject
   objectClass: organizationalRole
   cn: admin
   description: LDAP administrator

   # Groups, example.com
   dn: ou=Groups,dc=example,dc=com
   objectClass: organizationalUnit
   objectClass: top
   ou: Groups

   # users, Groups, example.com
   dn: cn=users,ou=Groups,dc=example,dc=com
   cn: users
   gidNumber: 100
   objectClass: posixGroup
   objectClass: top

   # Users, example.com
   dn: ou=Users,dc=example,dc=com
   objectClass: organizationalUnit
   objectClass: top
   ou: Users

   # Hosts, example.com
   dn: ou=Hosts,dc=example,dc=com
   objectClass: organizationalUnit
   objectClass: top
   ```

```
ou: Hosts

# Idmaps, example.com
dn: ou=Idmaps,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Idmaps

# samba, example.com
dn: cn=samba,dc=example,dc=com
cn: samba
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: SAMBA Access Account

# Aliases, example.com
dn: ou=Aliases,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Aliases

# nsspam, example.com
dn: cn=nsspam,dc=example,dc=com
cn: nsspam
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: NSS/PAM Access Account

# search result
search: 2
result: 0 Success

# numResponses: 11
# numEntries: 10
```

3. Add a new group record using the Please see SOLUTIONS/s_29/`groups.ldif` file included in the **solutions** directory.

```
# ldapadd -x -D "cn=admin,dc=example,dc=com" -W -H ldap://192.168.0.23 -f groups.ldif
```

```
 Enter LDAP Password:
adding new entry "cn=luser1,ou=Groups,dc=example,dc=com "
```

4. The **ldapsearch** command should now show a new group.

```
# ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub"objectclass=*"
```

```
 extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# example.com
dn: dc=example,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: example.com
dc: example
```

```
# admin, example.com
dn: cn=admin,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# Groups, example.com
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Groups

# users, Groups, example.com
dn: cn=users,ou=Groups,dc=example,dc=com
cn: users
gidNumber: 100
objectClass: posixGroup
objectClass: top

# Users, example.com
dn: ou=Users,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Users

# Hosts, example.com
dn: ou=Hosts,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Hosts

# Idmaps, example.com
dn: ou=Idmaps,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Idmaps

# samba, example.com
dn: cn=samba,dc=example,dc=com
cn: samba
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: SAMBA Access Account

# Aliases, example.com
dn: ou=Aliases,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Aliases

# nsspam, example.com
dn: cn=nsspam,dc=example,dc=com
cn: nsspam
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: NSS/PAM Access Account

# luser1, Groups, example.com
dn: cn=luser1,ou=Groups,dc=example,dc=com
cn: luser1
objectClass: posixGroup
objectClass: top
```

```
gidNumber: 999001
memberUid: luser1

# search result
search: 2
result: 0 Success

# numResponses: 12
# numEntries: 11
```

Notice in the output above the **numResponses** and **numEntries** have increased when we added the new group record.

5. Add a new user record using the Please see SOLUTIONS/s_29/`users.ldif` file included in the **solutions** directory.

```
# ldapadd -x -D "cn=admin,dc=example,dc=com" -W -H ldap://192.168.0.23 -f users.ldif
```

```
Enter LDAP Password:
adding new entry "cn=luser1,dc=example,dc=com"
```

6. The **ldapsearch** command should now show a new user.

```
# ldapsearch -x -H ldap://192.168.0.23 -b "dc=example,dc=com" -s sub"objectclass=*"
```

```
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# example.com
dn: dc=example,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: example.com
dc: example

# admin, example.com
dn: cn=admin,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# Groups, example.com
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Groups

# users, Groups, example.com
dn: cn=users,ou=Groups,dc=example,dc=com
cn: users
gidNumber: 100
objectClass: posixGroup
objectClass: top

# Users, example.com
dn: ou=Users,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
```

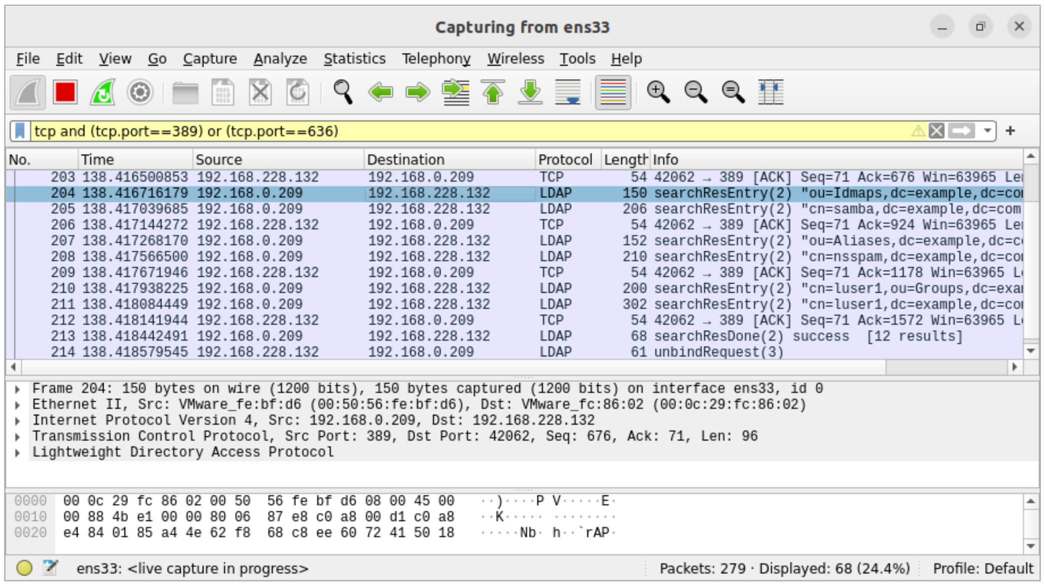
```
ou: Users

# Hosts, example.com
dn: ou=Hosts,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Hosts

# Idmaps, example.com
dn: ou=Idmaps,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Idmaps

# samba, example.com
dn: cn=samba,dc=example,dc=com
cn: samba
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: SAMBA Access Account

# Aliases, example.com
dn: ou=Aliases,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Aliases

# nsspam, example.com
dn: cn=nsspam,dc=example,dc=com
cn: nsspam
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: NSS/PAM Access Account

# luser1, Groups, example.com
dn: cn=luser1,ou=Groups,dc=example,dc=com
cn: luser1
objectClass: posixGroup
objectClass: top
gidNumber: 999001
memberUid: luser1

# luser1, example.com
dn: cn=luser1,dc=example,dc=com
uid: luser1
cn: luser1
givenName: luser1
sn: linux
homeDirectory: /home/users/luser1
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
uidNumber: 999001
gidNumber: 999001

# search result
search: 2
result: 0 Success

# numResponses: 13
# numEntries: 12
```

7. Install or verify wireshark is installed.

```
# yum install wireshark-gnome
# wireshark
```

8. Capture an ldapsearch command with wireshark, notice the data is clear text.

   Wireshark has excellent filters, use the filter "`tcp and (tcp.port==389) or (tcp.port==636)`" to have **wireshark** only display **LDAP** communication.


Capturing from ens33
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp and (tcp.port==389) or (tcp.port==636)
```

```
No.  Time          Source          Destination     Protocol Length Info
 203 138.416500853 192.168.228.132 192.168.0.209   TCP        54 42062 → 389 [ACK] Seq=71 Ack=676 Win=63965 Le
 204 138.416716179 192.168.0.209   192.168.228.132 LDAP      150 searchResEntry(2) "ou=Idmaps,dc=example,dc=co
 205 138.417039685 192.168.0.209   192.168.228.132 LDAP      206 searchResEntry(2) "cn=samba,dc=example,dc=com
 206 138.417144272 192.168.228.132 192.168.0.209   TCP        54 42062 → 389 [ACK] Seq=71 Ack=924 Win=63965 Le
 207 138.417268170 192.168.0.209   192.168.228.132 LDAP      152 searchResEntry(2) "ou=Aliases,dc=example,dc=c
 208 138.417566500 192.168.0.209   192.168.228.132 LDAP      210 searchResEntry(2) "cn=nsspam,dc=example,dc=co
 209 138.417671946 192.168.228.132 192.168.0.209   TCP        54 42062 → 389 [ACK] Seq=71 Ack=1178 Win=63965 L
 210 138.417938225 192.168.0.209   192.168.228.132 LDAP      200 searchResEntry(2) "cn=luser1,ou=Groups,dc=exa
 211 138.418084449 192.168.0.209   192.168.228.132 LDAP      302 searchResEntry(2) "cn=luser1,dc=example,dc=co
 212 138.418141944 192.168.228.132 192.168.0.209   TCP        54 42062 → 389 [ACK] Seq=71 Ack=1572 Win=63965 L
 213 138.418442491 192.168.0.209   192.168.228.132 LDAP       68 searchResDone(2) success  [12 results]
 214 138.418579545 192.168.228.132 192.168.0.209   LDAP       61 unbindRequest(3)
```

```
Frame 204: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface ens33, id 0
Ethernet II, Src: VMware_fe:bf:d6 (00:50:56:fe:bf:d6), Dst: VMware_fc:86:02 (00:0c:29:fc:86:02)
Internet Protocol Version 4, Src: 192.168.0.209, Dst: 192.168.228.132
Transmission Control Protocol, Src Port: 389, Dst Port: 42062, Seq: 676, Ack: 71, Len: 96
Lightweight Directory Access Protocol
```

```
0000  00 0c 29 fc 86 02 00 50  56 fe bf d6 08 00 45 00   ··)····P V·····E·
0010  00 88 4b e1 00 00 80 06  87 e8 c0 a8 00 d1 c0 a8   ··K····· ········
0020  e4 84 01 85 a4 4e 62 f8  68 c8 ee 60 72 41 50 18   ·····Nb· h··`rAP·
```

```
ens33: <live capture in progress>    Packets: 279 · Displayed: 68 (24.4%)    Profile: Default
```

Figure 29.2: **Wireshark LDAP packet trace**

Exercise 29.3: Centralized Authentication using LDAP for user authentication

In this exercise, we configure the client to use **LDAP** for centralized authentication.

Very Important

Starting with **CentOS-8** and **Ubuntu 20.04** the **System Security Services Daemon** or **sssd** is being used to manage access to remote directories and authentication. The use and configuration **sssd** is the same across the participating distributions.

1. Install required packages.

```
# apt install sssd sssd-ldap ldap-utils oddjob-mkhomedir
```

2. Create or update the **sssd** configuration to include the elements listed substituting your **ldap** server IP address in the **ldap_uri** field.

/etc/sssd/conf.d/00-sssd.conf

```
[sssd]
config_file_version = 2
domains = example.com
services = nss, pam,autofs

[domain/example.com]
enumerate = true
```

```
id_provider = ldap
autofs_provider = ldap
auth_provider = ldap
chpass_provider = ldap
ldap_uri = ldap://192.168.122.154/
ldap_search_base = dc=example,dc=com
ldap_id_use_start_tls = true
cache_credentials = True
ldap_tls_reqcert =allow
```

3. Verify and set the permissions for the **sssd.conf** file.

```
# chmod 600 /etc/sssd/conf.d/00-sssd.conf
# chown root.root /etc/sssd/conf.d/00-sssd.conf
```

4. Set up **oddjob** to automatically create home directories. Add **oddjob** to `/etc/pam.d/common-session` as in the following example:

/etc/pam.d/common-session.conf

```
session required        pam\_unix.so

session optional        pam\_oddjob\_mkhomedir.so

session optional        pam\_sss.so
```

5. Restart the services to pick up the changes.

```
# systemctl restart sssd oddjobd
# systemctl enable sssd oddjobd
```

6. Test the **ldap** server.

7. Verify the user information is available for user **luser1**

```
# getent passwd luser1
```

The response should be:

```
luser1:*:999001:999001:luser1:/home/users/luser1:
```

8. Test the user authorization is functioning for user **luser1**. The password should be password. There may not be a home directory available.

```
# ssh luser1@localhost
```

9. Verify the data stream is clear text.

   - Open a **wireshark** session filtering on ports 389 and 636.
   - While **wireshark** is capturing packets, log on and off as user luser1 via ssh to localhost.
   - Observe the output in **wireshark**

10. Disable the certificate validation.

Very Important

Disabling the certificate validation **NOT RECOMMENDED in production systems.**

Edit the /etc/sssd/sssd.conf file with your favorite editor. In the [domain/default] section add the line:

```
ldap_tls_reqcert = never
```

Restart the sssd service.

```
# systemctl restart sssd
```

11. Test the user information is being supplied by **LDAP** for **luser1**

```
# getent passwd luser1
```

12. Test the user authorization is functioning for **luser1**.

```
# ssh luser1@localhost
```

13. Verify the data stream to and from the **LDAP** server is encrypted.

    - Open a **wireshark** session filtering on ports 389 and 636.
    - While **wireshark** is capturing packets, log on and off as user luser1 via ssh to localhost.
    - Observe the output in **wireshark**.

14. Restore the original configuration when completed.

Chapter 30

Firewalls

30.1 Labs

Exercise 30.1: Installing firewalld

While most **Linux** distributions now have the **firewalld** package (which includes the **firewall-cmd** multi-purpose utility) available, it might not be installed on your system.

First you should check to see if it is already installed, with

```
$ which firewalld firewall-cmd
```

```
/usr/sbin/firewalld
/usr/bin/firewall-cmd
```

If you fail to find the program, then you need to install with one of the following, depending on your distribution in the usual way:

```
$ sudo dnf install firewalld
$ sudo zypper install firewalld
$ sudo apt install firewalld
```

If this fails, the **firewalld** package is not available for your distribution. In this case you will have to install from source.

To do this, go to https://fedorahosted.org/firewalld/ and you can get the **git** source repository, or you can easily download the most recent tarball.

Then you have to follow the common procedure for installing from source (using whatever the current version is):

```
$ tar xvf firewalld-0.3.13.tar.bz2
$ cd firewalld-0.3.13
$ ./configure
$ make
$ sudo make install
```

Note this source also has an **uninstall** target:

```
$ sudo make uninstall
```

in case you have regrets.

You will have to deal with any inadequacies that come up in the `./configure` step, such as missing libraries etc. When you install from a packaging system, the distribution takes care of this for you, but from source it can be problematic. If you have run the **Linux Foundation**'s **ready-for.sh** script on your system, you are unlikely to have problems.

Exercise 30.2: Examining firewall-cmd

We have only scratched the surface of how you can use the **firewalld** package. Almost everything is done by deploying **firewall-cmd** which is empowered to do a large variety of tasks, using options with very clear names.

To get a sense of this, there is really no substitute for just doing:

```
$ firewall-cmd --help
```

```
Usage: firewall-cmd [OPTIONS...]
....
Service Options
  --new-service=<service>
                       Add a new service [P only]
  --delete-service=<service>
                       Delete and existing service [P only]
....
```

which we will not reproduce here as it is 409 lines on an **RHEL 8** system.

For more detailed explanation of anything which piques your interest, do **man firewall-cmd** which explains things more deeply, and **man firewalld** which gives an overview, as well as a listing of other **man** pages that describe the various configuration files in `/etc`, and elucidate concepts such as **zones** and **services**.

Exercise 30.3: Adding Services to a Zone

Add the **http** and **https** services to the **public zone** and verify that they are currently listed.

Solution 30.3

```
$ sudo firewall-cmd  --zone=public --add-service=http
```

```
success
```

```
$ sudo firewall-cmd  --zone=public --add-service=https
```

```
success
```

```
$ sudo firewall-cmd --list-services --zone=public
```

```
dhcpv6-client http https ssh
```

Note if you had run

```
$ sudo firewall-cmd --reload
$ sudo firewall-cmd --list-services --zone=public
```

```
dhcpv6-client ssh
```

after adding the new services, they would disappear from the list! This curious behavior is because we did not include the `--permanent` flag when adding the services, and the `--reload` option reloads the known persistent services only.

Exercise 30.4: Using the firewall GUI

Each distribution has its own graphical interface for firewall administration. On **Red Hat**-based systems you can run **firewall-config**, on **Ubuntu** it is called **gufw**, and on **openSUSE** you can find it as part of **yast** on the graphical menu system.

We have concentrated on the command line approach simply because we want to be distribution-flexible. However, for most relatively simple firewall configuration tasks, you can probably do them efficiently with less memorization from the GUI.

Once you launch the firewall configuration GUI, do the previous exercise of adding **http** and **https** to the **public** zone, and verify that it has taken effect.

Make sure you take the time to understand the graphical interface.

Exercise 30.5: Determine your real IP address

In this exercise we will see the most common usage for **NAT**, consealing the `internal` or `private` addresses of devices behind a **firewall**.

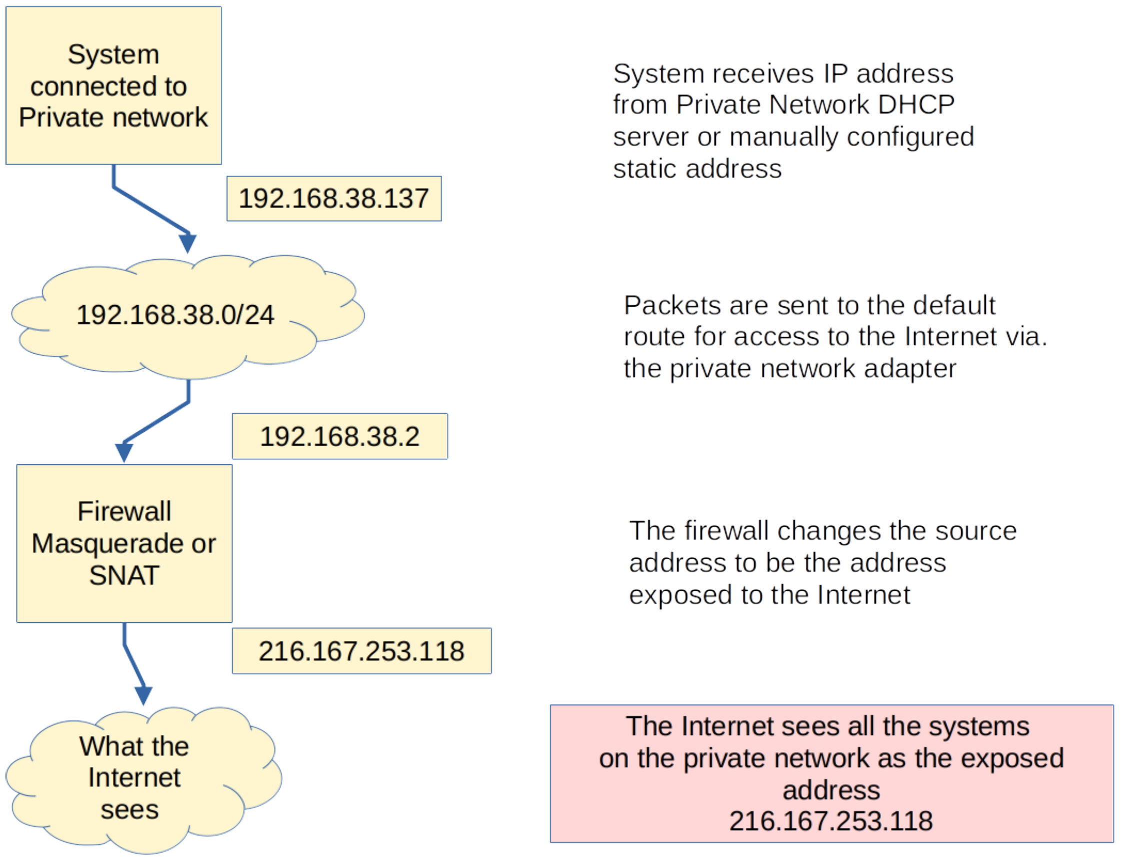

System connected to Private network
192.168.38.137
System receives IP address from Private Network DHCP server or manually configured static address
192.168.38.0/24
Packets are sent to the default route for access to the Internet via. the private network adapter
192.168.38.2
Firewall Masquerade or SNAT
The firewall changes the source address to be the address exposed to the Internet
216.167.253.118
What the Internet sees
The Internet sees all the systems on the private network as the exposed address 216.167.253.118

Figure 30.1: **SNAT or MASQUERADE example**

Using the above diagram as an guide determine the IP addresses in use in your system.

1. On your Linux VM, retrieve your network addresses. The **ip** command will work well, and the extra information not needed can be eliminated as shown below.

   ```
   $ sudo ip a | grep " inet "
   ```

   ```
   inet 127.0.0.1/8 scope host lo
   inet 192.168.38.137/24 brd 192.168.38.255 scope global dynamic noprefixroute ens33
   ```

   The addresses presented are the **loopback** and the IP address of the Virtual Machine.

2. The address of the VM is in the subnet of **192.168.24.0/24** and is a **non-routeable** address on the **internet**. See the link below for additional information on **Address Allocation for Private Internets**

https://datatracker.ietf.org/doc/html/rfc1918
To find the next IP address that will connect out client to the next hop, examine the **default route**.

```
$ sudo ip r
```

```
default via 192.168.38.2 dev ens33 proto dhcp metric 100
```

We can see our next hop is the system with the IP address of **192.168.38.2**.

3. The system **192.168.38.2** is unavailable for us to login to as it is the real firewall for this environment. We can extract how the world sees our system.
   Via command line:

```
$ sudo lynx -dump http://whatismyip.akamai.com  | awk '{print $1}'
```

```
216.167.253.118
```

   Or select one of many graphical options via a Goolge search for:
   what is my ip address

   Notice the address used to communicate with the outside world is not one of the private addresses and all the machines on the private network look like one IP Address to the outside world.
   The action of changing the private IP Addresses to the single address is an example of **Source Network Translation** or **Masquerade**

Chapter 31

System Init: systemd history and customization

31.1 Labs

Exercise 31.1: Modify an Existing Service with systemd

Often we want to modify the behavior of an existing service rather than creating a new one. **systemd** can handle this by completely replacing or disabling a built-in service, but also by only modifying certain parameters of an existing service, called a **drop-in** via an `.override` file or `.config` directory. An added advantage of using this method – package upgrades won't revert your changes.

We are going to modify the `Description` for the **unit** responsible for login (`getty`) on the first console (`tty1`) and then change it back.

We are assuming you are logged-in to your workstation either via the graphical interface or ssh. But a safety precaution, verify that you are not logged in on this terminal:

```
$ tty
```

Very Important

If you see the result of this command as `/dev/tty1`, it's probably best to skip this lab.

1. First, check the status of the `getty@tty1.service`:

   ```
   $ systemctl status getty@tty1.service
   ```

   ```
   $ systemctl status getty@tty1.service
   * getty@tty1.service - Getty on tty1
        Loaded: loaded (/lib/systemd/system/getty@.service; enabled; vendor preset: enabled)
        Active: active (running) since Thu 2023-03-09 12:58:42 EST; 2 months 7 days ago
          Docs: man:agetty(8)
                man:systemd-getty-generator(8)
                http://0pointer.de/blog/projects/serial-console.html
      Main PID: 621 (agetty)
         Tasks: 1 (limit: 4656)
   ```

```
       Memory: 200.0K
          CPU: 2ms
       CGroup: /system.slice/system-getty.slice/getty@tty1.service
               `-621 /sbin/agetty -o -p -- \u --noclear tty1 linux

Warning: some journal files were not opened due to insufficient permissions.
```

2. Next, let's change the description of the service. Copy the **Description=** line and make it say something different. In the example, we are changing the dynamic output to **first terminal**.

```
$ sudo systemctl edit getty@tty1.service
```

```
### Editing /etc/systemd/system/getty@tty1.service.d/override.conf
### Anything between here and the comment below will become the new contents of the file

[Unit]
Description=Getty on the first terminal

### Lines below this comment will be discarded

### /lib/systemd/system/getty@.service
# #  SPDX-License-Identifier: LGPL-2.1-or-later
# #
# #  This file is part of systemd.
# #
# #  systemd is free software; you can redistribute it and/or modify it
# #  under the terms of the GNU Lesser General Public License as published by
# #  the Free Software Foundation; either version 2.1 of the License, or
# #  (at your option) any later version.
```

3. Now, verify the contents of `/etc/systemd/system/getty@tty1.service.d/override.conf`
This should represent the change that you just made - just two lines with no newline at the end.

```
$ cat /etc/systemd/system/getty@tty1.service.d/override.conf
```

```
[Unit]
Description=Getty on the first terminal<YOUR PROMPT HERE>
```

4. Reload the daemon to pick up our changes, and the check the results:

```
$ sudo systemctl daemon-reload
$ systemctl status getty@tty1.service
```

```
* getty@tty1.service - Getty on the first terminal
     Loaded: loaded (/lib/systemd/system/getty@.service; enabled; vendor preset: enabled)
    Drop-In: /etc/systemd/system/getty@tty1.service.d
             `-override.conf
     Active: active (running) since Thu 2023-03-09 12:58:42 EST; 2 months 7 days ago
       Docs: man:agetty(8)
             man:systemd-getty-generator(8)
             http://0pointer.de/blog/projects/serial-console.html
   Main PID: 621 (agetty)
      Tasks: 1 (limit: 4656)
     Memory: 200.0K
        CPU: 2ms
     CGroup: /system.slice/system-getty.slice/getty@tty1.service
             `-621 /sbin/agetty -o -p -- \u --noclear tty1 linux

Warning: some journal files were not opened due to insufficient permissions.
```

5. Finally, remove the file we created to bring things back to normal.

```
$ sudo rm  /etc/systemd/system/getty@tty1.service.d/override.conf
$ systemctl status getty@tty1.service
```

```
   * getty@tty1.service - Getty on tty1
        Loaded: loaded (/lib/systemd/system/getty@.service; enabled; vendor preset: enabled)
        Active: active (running) since Thu 2023-03-09 12:58:42 EST; 2 months 7 days ago
          Docs: man:agetty(8)
                man:systemd-getty-generator(8)
                http://0pointer.de/blog/projects/serial-console.html
      Main PID: 621 (agetty)
         Tasks: 1 (limit: 4656)
        Memory: 200.0K
           CPU: 2ms
        CGroup: /system.slice/system-getty.slice/getty@tty1.service
                `-621 /sbin/agetty -o -p -- \u --noclear tty1 linux

   Warning: some journal files were not opened due to insufficient permissions.
```

Chapter 32

Backup and Recovery Methods

32.1 Labs

Exercise 32.1: Using tar for Backup

1. Create a directory called `backup` and in it place a compressed **tar** archive of all the files under `/usr/include`, with the highest level directory being `include`. You can use any compression method (**gzip**, **bzip2** or **xzip**).
2. List the files in the archive.
3. Create a directory called `restore` and unpack and decompress the archive.
4. Compare the contents with the original directory the archive was made from.

Solution 32.1

1.
```
$ mkdir /tmp/backup
$ cd /usr ; tar zcvf /tmp/backup/include.tar.gz include
$ cd /usr ; tar jcvf /tmp/backup/include.tar.bz2 include
$ cd /usr ; tar Jcvf /tmp/backup/include.tar.xz include
```
   or
```
$ tar -C /usr -zcf include.tar.gz include
$ tar -C /usr -jcf include.tar.bz2 include
$ tar -C /usr -Jcf include.tar.xz include
```
   Notice the efficacy of the compression between the three methods:
```
$ du -sh /usr/include
```
```
55M        /usr/include
```

2.
```
$ ls -lh include.tar.*
```
```
c8:/tmp/backup>ls -lh
total 17M
-rw-rw-r-- 1 coop coop 5.3M Jul 18 08:17 include.tar.bz2
-rw-rw-r-- 1 coop coop 6.7M Jul 18 08:16 include.tar.gz
-rw-rw-r-- 1 coop coop 4.5M Jul 18 08:18 include.tar.xz
```

3.
```
$ tar tvf include.tar.xz
```

```
qdrwxr-xr-x root/root         0 2014-10-29 07:04 include/
-rw-r--r-- root/root     42780 2014-08-26 12:24 include/unistd.h
-rw-r--r-- root/root       957 2014-08-26 12:24 include/re_comp.h
-rw-r--r-- root/root     22096 2014-08-26 12:24 include/regex.h
-rw-r--r-- root/root      7154 2014-08-26 12:25 include/link.h
.....
```

Note it is not necessary to give the `j`, `J`, or `z` option when decompressing; **tar** is smart enough to figure out what is needed.

4.
```
$ cd .. ; mkdir restore ; cd restore
$ tar xvf ../backup/include.tar.bz2
```

```
include/
include/unistd.h
include/re_comp.h
include/regex.h
include/link
.....
$ diff -qr include /usr/include
```

Exercise 32.2: Using rsync for Backup

1. Using **rsync**, we will again create a complete copy of `/usr/include` in your backup directory:

```
$ rm -rf include
$ rsync -av /usr/include .
```

```
sending incremental file list
include/
include/FlexLexer.h
include/_G_config.h
include/a.out.h
include/aio.h
.....
```

2. Let's run the command a second time and see if it does anything:

```
$ rsync -av /usr/include .
```

```
sending incremental file list

sent 127398 bytes  received 188 bytes  255172.00 bytes/sec
total size is 41239979  speedup is 323.23
```

3. One confusing thing about **rsync** is you might have expected the right command to be:

```
$ rsync -av /usr/include include
```

```
sending incremental file list
...
```

However, if you do this, you'll find it actually creates a new directory, `include/include`!

4. To get rid of the extra files you can use the `--delete` option:

```
$ rsync -av --delete /usr/include .
```

```
sending incremental file list
include/
deleting include/include/xen/privcmd.h
deleting include/include/xen/evtchn.h
....
deleting include/include/FlexLexer.h
deleting include/include/

sent 127401 bytes  received 191 bytes  85061.33 bytes/sec
total size is 41239979  speedup is 323.22
```

5. For another simple exercise, remove a subdirectory tree in your backup copy and then run **rsync** again with and without the `--dry-run` option:

```
$ rm -rf include/xen
$ rsync -av --delete --dry-run /usr/include .
```

```
sending incremental file list
include/
include/xen/
include/xen/evtchn.h
include/xen/privcmd.h

sent 127412 bytes  received 202 bytes  255228.00 bytes/sec
total size is 41239979  speedup is 323.16 (DRY RUN)
```

```
$ rsync -av --delete  /usr/include .
```

6. A simple script with a good set of options for using **rsync**:

script using rsync

```
#!/bin/sh
set -x

rsync --progress -avrxH --delete $*
```

which will work on a local machine as well as over the network. Note the important `-x` option which stops **rsync** from crossing filesystem boundaries.

Extra Credit

For more fun, if you have access to more than one computer, try doing these steps with source and destination on different machines.

Chapter 33

Linux Security Modules

33.1 Labs

Exercise 33.1: SELinux: Contexts

Please Note

This exercise can only be performed on a system (such as **RHEL**) where **SELinux** is installed. While it is possible to install on **Debian**-based distributions, such as **Ubuntu**, it is not the easiest task and it is not often done.

1. Verify **SELinux** is enabled and in **enforcing** mode, by executing **getenforce** and **sestatus**. If not, edit `/etc/selinux/config`, reboot, and check again.
2. Install the **httpd** package (if not already present) which provides the **Apache** web server, and then verify that it is working:

```
$ sudo dnf install httpd
$ sudo systemctl start httpd
$ elinks http:/localhost
```

   (You can also use **lynx** or **elinks** etc. as the browser, or use your graphical browser such as **firefox** or **chrome**, in this and succeeding steps.)
3. As superuser, create a small file in `/var/www/html`:

```
$ sudo sh -c "echo file1 > /var/www/html/file1.html"
```

4. Verify you can see it:

```
$ elinks -dump http://localhost/file1.html
```

```
   file1
```

   Now create another small file in **root**'s home directory and **move** it to `/var/www/html`. (Do not copy it, move it!) Then try and view it:

```
$ sudo cd /root
$ sudo sh -c "echo file2 > file2.html"
$ sudo mv file2.html /var/www/html
$ elinks -dump http://localhost/file2.html
```

```
                              Forbidden
```

```
You don't have permission to access /file2.html on this server.
```

5. Examine the security contexts:

```
$ cd  /var/www/html
$ ls -Z file*html
```

```
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 file1.html
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file2.html
```

6. Change the offending context and view again:

```
$ sudo chcon -t httpd_sys_content_t file2.html
$ elinks http://localhost/file2.html
```

```
file2
```

Exercise 33.2: Exploring apparmor security

Please Note

This exercise can only be performed on a system (such as **Ubuntu**) where **AppArmor** is installed.

The below was tested on **Ubuntu**, but should work on other **AppArmor**-enabled systems, such as **openSUSE**, where the **apt** commands should be replaced by **zypper**.

On **Ubuntu**, the **/bin/ping** utility runs with SUID enabled. For this exercise, we will copy **ping** to **ping-x** and adjust the capabilities so the program functions.

Then we will build an **AppArmor** profile, install and verify that nothing has changed. Modifying the **AppArmor** profile and adding capabilities will allow the program more functionality.

1. Make sure all necessary packages are installed:

```
student@ubuntu:~$ sudo apt install apparm*
```

2. Create a copy of **ping** (called **ping-x**) and verify it has no initial special permissions or capabilities. Furthermore, it cannot work when executed by **student**, a normal user:

```
student@ubuntu:~$ sudo cp /bin/ping /bin/ping-x
student@ubuntu:~$ sudo ls -l /bin/ping-x

-rwxr-xr-x 1 root root 64424 Oct 17 10:12 /bin/ping-x

student@ubuntu:~$ sudo getcap /bin/ping-x
student@ubuntu:~$

student@ubuntu:~$ ping-x -c3 -4 127.0.0.1
ping: socket: Operation not permitted
```

3. Set the **capabilities** and re-try **ping-x**:

```
student@ubuntu:~$ sudo setcap cap_net_raw+ep  /bin/ping-x

student@ubuntu:~$ ping-x  -c3 -4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.092 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.093 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.086 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2034ms
rtt min/avg/max/mdev = 0.086/0.090/0.093/0.008 ms
```

The modified **ping-x** program now functions normally.

4. Verify there is no pre-existing **AppArmor** profile for **ping-x**, but there is a profile for **ping**. Determine the status of the current **ping** program:

```
student@ubuntu:˜$ sudo aa-status
```

The output from **aa-status** is long, so we can **grep** for the interesting lines:

```
student@ubuntu:˜$ sudo aa-status | grep -e "ˆ[[:alnum:]]" -e ping

apparmor module is loaded.
87 profiles are loaded.
51 profiles are in enforce mode.
   ping
36 profiles are in complain mode.
17 processes have profiles defined.
6 processes are in enforce mode.
11 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

We can see **ping** has a profile that is loaded and enabled for enforcement.

5. Next we will construct a new profile for **ping-x**. This step requires two terminal windows.
The first window (**window1**)will be running the **aa-genprof** command. This will generate a **AppArmor** profile by scanning `/var/log/syslog` for **AppArmor** errors.
The second window (**window2**) will be used to run **ping-x**. (See the **man** page for **aa-genprof** for additional information.)
In **window1**:

```
student@ubuntu:˜$ sudo aa-genprof /bin/ping-x
Writing updated profile for /bin/ping-x.
Setting /bin/ping-x to complain mode.

Before you begin, you may wish to check if a
profile already exists for the application you
wish to confine. See the following wiki page for
more information:
http://wiki.apparmor.net/index.php/Profiles

Please start the application to be profiled in
another window and exercise its functionality now.

Once completed, select the "Scan" option below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.

Profiling: /bin/ping-x

[(S)can system log for AppArmor events] / (F)inish
```

In **window2**:

```
student@ubuntu:˜$ ping-x  -c3 -4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.099 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.120 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.114 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2041ms
rtt min/avg/max/mdev = 0.099/0.111/0.120/0.008 ms
```

In **window1**:
The command `ping-x` has completed, we must now instruct **aa-genprof** to scan for the required information to be added to the profile. It may require several **scans** to collect all of the information for the profile.
Enter `S` to scan:

```
Reading log entries from /var/log/syslog.
Updating AppArmor profiles in /etc/apparmor.d.
Complain-mode changes:

Profile:    /bin/ping-x
Capability: net_raw
Severity:   8

 [1 - capability net_raw,]
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish
```

Enter A to allow the capability:

```
Adding capability net_raw, to profile.

Profile:        /bin/ping-x
Network Family: inet
Socket Type:    raw

 [1 - network inet raw,]
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish
```

Enter A to allow the network family:

```
Adding network inet raw, to profile.

Profile:        /bin/ping-x
Network Family: inet
Socket Type:    dgram

 [1 - #include <abstractions/nameservice>]
  2 - network inet dgram,
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish
```

Enter A to add the socket type datagram to the profile:

```
Adding #include <abstractions/nameservice> to profile.

= Changed Local Profiles =

The following local profiles were changed. Would you like to save them?

 [1 - /bin/ping-x]
(S)ave Changes / Save Selec(t)ed Profile / [(V)iew Changes] / View Changes b/w (C)lean profiles / Abo(r)t
```

Enter S to save the new profile:

```
Writing updated profile for /bin/ping-x.

Profiling: /bin/ping-x

[(S)can system log for AppArmor events] / (F)inish
```

Enter F to finish:

```
Setting /bin/ping-x to enforce mode.

Reloaded AppArmor profiles in enforce mode.

                Please consider contributing your new profile!
See the following wiki page for more information:
http://wiki.apparmor.net/index.php/Profiles

Finished generating profile for /bin/ping-x.
```

6. View the created profile, which has been stored in `/etc/appamor.d/bin.ping-x`.

```
student@ubuntu:~$ sudo cat /etc/apparmor.d/bin.ping-x
# Last Modified: Tue Oct 17 11:30:47 2017
#include <tunables/global>
```

```
/bin/ping-x {
  #include <abstractions/base>
  #include <abstractions/nameservice>

  capability net_raw,

  network inet raw,

  /bin/ping-x mr,
  /lib/x86_64-linux-gnu/ld-*.so mr,

}
```

7. The **aa-genproc** utility installs and activates the new policy so it should be ready to use, and the policies can be reloaded on demand with the `systemctl reload apparmor` command. To avoid any potential issues, and verify the changes will survive, reboot the system.
   Once the system has restarted, as the user `student`, verify **ping-x** still functions with the new profile enabled. Ping the localhost by ip address:

```
student@ubuntu:~$ ping-x -c3 -4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.095 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 0.043/0.065/0.095/0.021 ms
```

8. This should work as expected. The profile is very specific, and **AppArmor** will not allow functionality outside of the specified parameters. To verify **AppArmor** is protecting this application, try to ping the **IPV6** localhost address.
   This should fail:

```
student@ubuntu:~$ ping-x -c3  -6 ::1
ping: socket: Permission denied
```

   (Note, the `-6` option means use only **IPv6** and `::1` is the local host in **IPv6**.)
   The output indicates there is a socket issue. If the system log is examined it will be discovered that our **ping-x** program has no access to **IPv6** within **AppArmor**:

```
766:104): apparmor="DENIED" operation="create" profile="/bin/ping-x"
pid=2709 comm="ping-x" family="inet6" sock_type="raw" protocol=58
requested_mask="create" denied_mask="create
```

9. To correct this deficiency, re-run **aa-genprof** as we did earlier, and in **window2**, ping the **IPv6** loopback and append the additional options

Chapter 34

System Rescue

34.1 Labs

Exercise 34.1: Preparing to use Rescue/Recover Media

Very Important

In the following exercises we are going to deliberately damage the system and then recover through the use of rescue media. Thus, it is obviously prudent to make sure you can indeed boot off the rescue media before you try anything more ambitious.

So first make sure you have rescue media, either a dedicated rescue/recovery image, or an install or Live image on either an optical disk or usb drive.

Boot off it and make sure you know how to force the system to boot off the rescue media (you are likely to have to fiddle with the **BIOS** settings), and when the system boots, choose rescue mode.

Please Note

If you are using a virtual machine, the procedure is logically the same with two differences:

- Getting to the **BIOS** might be difficult depending on the hypervisor you use. Some of them require very rapid keystrokes, so read the documentation and make sure you know how to do it.
- You can use a physical optical disk or drive, making sure the virtual machine settings have it mounted, and if it is **USB** you may have some other hurdles to make sure the virtual machine can claim the physical device. It is usually easier to simply connect a `.iso` image file directly to the virtual machine.

If you are working with a virtual machine, obviously things are less dangerous, and if you are afraid of corrupting the system in an unfixable way, simply make a backup copy of the virtual machine image before you do these exercises, you can always replace the image with it later.

Very Important

Do not do the following exercises unless you are sure you can boot your system off rescue/recovery media!

Exercise 34.2: Recovering from a Corrupted GRUB Configuration

1. Edit your **GRUB** configuration file (`/boot/grub/grub.cfg`, `/boot/grub2/grub.cfg` or `/boot/grub/grub.conf`), and modify the `kernel` line by removing the first character of the value in the field named `UUID`. Take note of which character you removed, you will replace it in rescue mode. (If your root filesystem is identified by either label or hard disk device node, make an analogous simple change.) Keep a backup copy of the original.

   On a BLSCFG System

   You can corrupt the command line by editing `/etc/grub2/grubenv` instead.

2. Reboot the machine. The system will fail to boot, saying something like `No root device` was found. You will also see that a `panic` occurred.
3. Insert into your machine the **installation** or **Live DVD** or **CD** or **USB** drive (or network boot media) if you have access to a functioning installation server). Reboot again. When the boot menu appears, choose to enter rescue mode.
4. As an alternative, you can try selecting a rescue image from the **GRUB** menu; most distributions offer this. You'll get the same experience as using rescue media, but it will not always work. For example, if the root filesystem is damaged it will be impossible to do anything.
5. In rescue mode, agree when asked to search for filesystems. If prompted, open a shell, and explore the rescue system by running utilities such as **mount** and **ps**.
6. Repair your broken system by fixing your **GRUB** configuration file, either by editing it or restoring from a backup copy.
7. Type `exit` to return to the installer, remove the boot media, and follow the instructions on how to reboot. Reboot your machine. It should come up normally.

Exercise 34.3: Recovering from Password Failure

1. As root (not with **sudo**), change the root password. We will pretend we don't know what the new password is.
2. Log out and try to login again as root using the old password. Obviously you will fail.
3. Boot using the rescue media, and select `Rescue` when given the option. Let it mount filesystems and then go to a command line shell.
4. Go into your **chroot**-ed environment (so you have normal access to your systems):

   ```
   $ chroot /mnt/sysimage
   ```

   and reset the root password back to its original value.
5. Exit, remove the rescue media, and reboot, you should be able to login normally now.

Exercise 34.4: Recovering from Partition Table Corruption

Very Important

This exercise is dangerous and could leave to an unusable system. Make sure you really understand things before doing it

Please Note

The following instructions for an **MBR** system. if you have **GPT** you need to use **sgdisk** with the `--backup-file` and `--load-backup` options as discussed in the partitioning chapter

1. Login as root and save your **MBR**:

   ```
   $ dd if=/dev/sda of=/root/mbrsave bs=446 count=1
   ```

   ```
   1+0 records in
   1+0 records out
   446 bytes (446 B) copied, 0.00976759 s, 45.7 kB/s
   ```

   Be careful: make sure you issue the exact command above and that the file saved has the right length:

   ```
   $ sudo ls -l /root/mbrsave
   ```

   ```
   -rw-r--r-- 1 root root 446 Nov 12 07:54 mbrsave
   ```

2. Now we are going to obliterate the **MBR** with:

   ```
   $ dd if=/dev/zero of=/dev/sda bs=446 count=1
   ```

   ```
   1+0 records in
   1+0 records out
   446 bytes (446 B) copied, 0.000124091 s, 3.6 MB/s
   ```

3. Reboot the system; it should fail.
4. Reboot into the rescue environment and restore the **MBR**:

   ```
   $ dd if=/mnt/sysimage/root/mbrsave of=/dev/sda bs=446 count=1
   ```

5. Exit from the rescue environment and reboot. The system should boot properly now.

Exercise 34.5: Recovering Using the Install Image

Please Note

This exercise has been specifically written for **Red Hat**-based systems. You should be able to easily construct the appropriate substitutions for other distribution families.

1. Remove the **zsh** package (if it is installed!):

   ```
   s$ dnf remove zsh
   ```

   or

   ```
   $ rpm -e zsh
   ```

   Note we have chosen a package that generally has no dependencies to simplify matters. If you choose something that does, you will have to watch your step in the below so that anything else you remove you reinstall as needed as well.
2. Boot into the rescue environment.
3. Re-install (or install) **zsh** from within the rescue environment. First, mount the install media at `/mnt/source`:

   ```
   $ mount /dev/cdrom /mnt/source
   ```

   Then reinstall the package:

   ```
   $ rpm -ivh --force --root /mnt/sysimage /mnt/source/Packages/zsh*.rpm
   ```

   The `--force` option tells **rpm** to use the source directory in determining dependency information etc. Note that if the install image is much older than your system which has had many updates the whole procedure might collapse!
4. Exit and reboot.
5. Check that **zsh** has been reinstalled:

   ```
   $ rpm -q zsh
   ```

```
zsh-5.0.2-7.el7.x86_64
```

6. `$ zsh`

```
....
[coop@q7]/tmp/LFS201%
```