Linux Reference

Version 1.2.4
Will Smith
`minkcv@gmail.com`

March 29, 2016

Preface

I wrote this reference for two reasons. One was to learn LaTeX, the typesetting system that makes this document look good. The second was to consolidate all the Linux knowledge that I found online when searching for answers to specific questions and wanted to be able to look up in an on-disk reference.

I should note that I am not responsible for any damage incurred from use of this document. I will allow copying and distribution of this document without modification. Any corrections should be sent to the email listed above.

This document is not a Linux tutorial, tour of the Linux file system, or comprehensive reference. Again, it is a collection of information that I found online and wanted in one place. This document may be useful to intermediate users.

I also have decided not to cite any works as most of this information comes from random anonymous posters on non academic forums.

Contents

1. Disks and Media Managment

WARNING: Be absolutely sure you know what you are doing before executing commands that write to disks such as dd. I am not responsible for lost data.

1.1 Listing Disks and Information

1.1.1 Showing All Devices, Partitions and Potential Mount Points

The command `lsblk` lists devices, partitions, sizes, and if a disk is mounted and where.

1.1.2 Finding UUIDs and Differentiating Volumes

The command `sudo blkid` will list the UUIDs of devices as well as the file system of partitions. This command also lists labels of disks which can be helpful in differentiating between disks of the same size. Running this command without root or sudo will not work.

1.1.3 Determining Free Space

The command `df` lists free and used space on partitions. It also lists mount points. Running `df -h` will use human readable sizes (4.0K rather than 4096).

1.2 Mounting Partitions

For this examples in this and related sections /dev/sdX and /dev/sdXN will be used to refer to drives and partitions. Also, /media/mountpoint/ may be used to refer to where volumes are mounted. Make sure to check ownership of mountpoints. Most commands in this section should be run as root or with sudo.

1.2.1 Mounting ext3/ext4 Partitions

After you know which partition you want to mount you can run `mount /dev/sdXN /media/mountpoint` to mount it.

1.2.2 Mounting ntfs Partitions

Running `mount /dev/sdXN /media/mountpoint` should work. If not, run `mount -t ntfs /dev/sdXN /media/mountpoint` to mount it. You may need to install the ntfs-3g package. This can be done on ubuntu with `sudo apt-get install ntfs-3g`.

Mounting with Permissions

If mounting an ntfs (or FAT32) partition in the above way does not allow you to write to it or chmod it (even as root) then it needs to be mounted with different permissions. Run `mount -o umask=0000 /dev/sdXN /media/mountpoint` to mount the volume and have all files and directories in the volume (as well as the folder the volume is mounted in) be readable, writeable, and executable by everyone. An alternative to this is to add the partition/device to /etc/fstab with the ”user” option so that non root users can mount the partition.

1.2.3 Mounting an Image

To mount an iso file run `mount -o loop disk.iso /media/mountpoint` to mount an image called disk.iso. This works for other raw data besides iso files as well.

1.2.4 Mounting a RAM Disk

A RAM disk is storage space in the computers random access memory. It will no longer exist if the computer is shut down, restarted, or power is interrupted. However, this storage is faster to read and write which may be useful in some scenarios. Make sure not to put too much data here as the computer uses the RAM to run the operating system and programs.

Without a Defined Size

Store files and directories in /dev/shm/.

With a Defined Size

Run `mount -t tmpfs -o size=1024m tmpfs /mount/point` to create a fixed size RAM disk.

1.3 Creating and Formatting Partitions

1.3.1 Creating a Partition

To create a partition on /dev/sdX run `fdisk /dev/sdX`. Enter `n` to create a new partition. Choose primary or extended. Disks have a maximum of four primary partitions. Extended partitions cannot be booted and should be used if four partitions is not enough. Most of the defaults will be sufficient for the rest of the options. Make sure to exit with `w` to write the changes. You can exit with `q` if you wish to abort.

1.3.2 Formatting a Partition

Formatting with ext3/ext4

Run `mkfs.ex4 /dev/sdXN` to create an ext4 partition with number N on device X. The command is the same except with `mkfs.ext3` for the ext3 file system.

Formatting with ntfs

Run `mkfs.ntfs /dev/sdXN` to create an ntfs partition. You may need to install the ntfs-3g package.

1.4 Using dd to Write to Disks and Partitions

Most commands with dd require root or sudo. The speed of some commands may be increased by changing the block size. Append `bs=1M` or the value of your choosing to the dd command. Note that the largest block size is not necessarily the fastest and using the wrong block size on some media may cause poor performance. Also note that the destination partition or disk should be of equal or greater size to that of the source partition.

1.4.1 Writing an Image to a Disk

Run `dd if=/path/image.img of=/dev/sdX` to write image.img to /dev/sdX. Note that the file does not need to end in .img.

1.4.2 Copying a Partition to Another Partition

Copy an entire partition to another partition with `dd if=/dev/sdXN of=/dev/sdYM conv=noerror,sync`. The noerror option causes dd to continue writing even if read errors occur. The sync option fills input blocks with zeroes if there were read errors.

1.4.3 Copying an Entire Disk to Another Disk

Run `dd if=/dev/sdX of=/dev/sdY conv=noerror,sync` to copy an entire disk (all partitions) to another disk.

1.4.4 Copying an Entire Disk to a File

Run `dd if=/dev/sdX of=/path/file.img conv=noerror,sync` to copy an entire disk to a file.

1.4.5 Showing the Progress of dd with pv

The command pv stands for pipe viewer and can be used to create a progress bar for some operations.

Using pv with dd When Copying Disks or Partitions

Run `dd if=/dev/sdX | pv | dd of=/dev/sdY` to copy /dev/sdX to /dev/sdY and show progress. You may need to install pv. If you want to add options to dd such as `bs=1M` they should be in specified in the first dd.

Using pv with dd When Copying a File to a Disk or Partition

Run `pv /path/file.img | dd of=/dev/sdX` to copy a file to a disk and show the progress.

1.5 Using /etc/fstab

The fstab file located in /etc/ is a file system table that is used to configure how partitions should be mounted. It is not entirely necessary to mount partitions this way but it can make the process easier. You must be sudo or root to edit /etc/fstab.

1.5.1 Columns of the File

Column 1: File System

The first column specifies device either by /dev/sdXN, LABEL or UUID. Drives specified by UUID look like `UUID=12ab1234-a12a-1234-1a23-1a23456789a1`. Drives specified by LABEL look like `LABEL=external`. Drive label can be found by running `lsblk -f` or `blkid` as root.

Column 2: Mount Point

The second column specifies mount point such as / or /media/mountpoint.

Column 3: File System Type

The third column specifies file system type such as ext3, ext4, ntfs, or swap. It is possible to set this field to `auto` to have it determined automatically. This is useful for some removable media. The following options are disregarded if the file system type is swap

Column 4: Options

The following are common options:

- **fmask, dmask, and umask**: changes the permissions for files, directories, and both respectively

- **auto and noauto**: specifies if the device should be mounted at startup or not
- **user and nouser**: specifies if non root users can mount the device or not
- **exec and noexec**: specifies if executables can be run from the disk or not
- **suid and nosuid**: specifies if allow set-user-identifier or set-group identifier bits take effect – these bits allow an executing program to change its user and group based privileges
- **dev and nodev**: specifies if non root users can create device nodes in the filesystem
- **ro**: mount the file system read only
- **rw**: mount the file system read write
- **sync and async**: specifies if changes should be written synchronously or asynchronously – synchronous means that changes are written immediately when you execute a command such as cp or mv
- **defaults**: uses the following options: rw, suid, dev, exec, auto, nouser, async

Column 5: Dump

Dump is a program for backing up drives. A value of zero (0) specifies that the file system should not be backed up by dump. If the value is one (1) then dump will make a backup. Possible values are 0 and 1.

Column 6: Pass or fsck

The sixth column indicates if and in what order the file systems should be checked by fsck. A value of zero (0) specifies that the file system should not be checked. Possible values are 0, 1, and 2. The root file system should have a value of 1.

An Example /etc/fstab

```
# <file system>   <dir>         <type>     <options>              <dump>  <pass>
/dev/sda1         /             ext4       defaults               0       1
LABEL=windows     /media/drive2 ntfs       defaults, umask=0022   0       0
/dev/sdb1         none          ext4       defaults               0       2
/dev/sda3         none          swap       defaults               0       0
```

1.6 Recovering Lost Data and Disk Health

1.6.1 Recovering Lost Data

testdisk

Run `testdisk`. After selecting the logging behavior, select the drive to operate on. Then select the partition table type. Testdisk may show a hint with the type it has detected. After selecting the type, select the Advanced option and the Undelete option. Use ”c” to copy the file highlighted with the cursor and ”C” once you have selected the directory to copy it to.

ddrescue

Run `ddrescue -d -r<retry-times> <source-disk> <destination> <log>` to copy <source-disk> to the <destination> and write the log to <log>. The -d option disables the operating system cache of the disk and <retry-times> is the number of times ddrescue will retry to read bad sectors. If the drive makes bad noises then keep the <retry-times> low because repeatedly retrying on a bad sector could break the entire disk. The destination can be a drive, partition, or file.

An example ddrescue command:
```
sudo ddrescue -d -r3 /dev/sdc  /disk.raw  /disk.log
```

1.6.2 Disk Health Using SMART

The command `smartctl` analyzes the health of disks and must be run as root or with sudo. Run `smartctl -d <type> -i <device>` where <type> is ata, scsi, or sat and <device> is the drive such as /dev/sda to show drive information. Run with the -a flag instead of -i to run an overall-health assesment.

An example smartctl command looks like:
```
sudo smartctl -d sat -a /dev/sda
```

2. Archiving and Compressing Files

2.1 Putting Multiple Files Into a tar File

Run `tar -cvf files.tar /path/directory/` to create a tar file with everything in directory. Remove the `v` flag to avoid printing out a list of files as they are added to the archive.

2.2 Compression

2.2.1 gzip

Add the `z` flag to the above arguments for tar (and add .gz for conveniece) to compress the archive with gzip. The resulting command looks like `tar -cvzf files.tar.gz /path/directory/`.

2.2.2 bz2

Add the `j` flag to tar (and add the .bz2 extension) to use bz2 compression. It will take longer than gzip but will result in a smaller file.

2.2.3 Other Compression Algorithms

Other compression algorithms such as lzma can be used. Check the tar manpages for a list. Some may need additional packages to be installed.

2.3 Extraction

2.3.1 Extracting a tar, tar.gz, or tar.bz2

Run `tar -xvf files.tar` to extract files into the current directory or `tar -xvf files.tar -c /path/directory/` to extract files into /path/directory/. The syntax is the same with tar.gz or tar.bz2 except if the filename ends in .gz or .bz2.

2.4 Splitting Files Into Multiple Parts

2.4.1 Splitting a File

Run `tar -czpvf - /path/file | split -d -b 1024M - tardisk` to split /path/file into 1024M (1G) files. This will create a bunch of files named `tardiskNM` where NM starts at 00 and increases with each part.

2.4.2 Unsplitting a File

Run `cat tardisk* | tar -xzpvf -` to reconstruct the original file.

3. Networking

3.1 Local Network Configuration

3.1.1 Determining Current Configuration

- Run `ifconfig` to show a list of interfaces and related information such as ip addresses and masks.
- Run `ip addr` to similar information as `ifconfig`
- Run `route -n` to show the gateway for the interfaces.

3.1.2 Enabling and Disabling an Interfaes

These commands must be run as root or with sudo. Run `ifconfig <interface> <state>` where <interface is your interface such as eth0 or wlan0 and <state> is either `up` or `down`.

Setting a Hardware Address (MAC Address) with ifconfig

Make sure your interface is down before trying to change its hardware address. Run `ifconfig <interface> hw ether <mac-address>` where <interface> is the desired interface and <mac-address> is the desired new mac address in the form `0a:1b:2c:3d:4e:5f`.

3.1.3 IP Configuration with /etc/network/interfaces

For each interface the file has a number of settings, some optional and some mandatory. Each interface starts with a line like `iface <interface> inet <type>` where <type> is `static`, `dhcp`, or `manual`. Each interface may be preceeded by an `auto <interface>` line that causes <interface> to be started at boot. The options for the interfaces should be indented.

Dynamic Configuration (DHCP)

A dynamic configuration starts with `iface <interface> inet dhcp`

Static Configuration

A static configuration starts with `iface <interface> inet static` and should then be followed by the address, netmask, and gateway lines. Some additional lines may be specified.

Manual Configuration

A manual configuration starts with `iface <interface inet manual` is configured through other programs, and is not described here.

Options for Configurations

Mandatory for static configurations

- `ipaddress <ip-address>`
- `netmask <mask>`
- `gateway <gateway>`

Optional for static configurations

- `broadcast <broadcast>`

Optional for static and dynamic configurations

- `hwaddress ether <mac-address>`
- `dns-nameservers <ip-address> <ip-address-2>`

More options such as `pre-up` and `bridge_ports` are available but not listed in this reference.

Example /etc/network/interfaces

Dynamic Configuration:

```
auto eth0
iface eth0 inet dhcp
```

Static Configuration

```
auto eth0
iface eth0 inet static
   address 192.168.1.12
   netmask 255.255.255.0
   gateway 192.168.1.1
```

3.2 Uncomplicated Firewall (ufw)

Uncomplicated Firewall is a program that simplifies allowing, denying, and rerouting network activities. It modifies iptables which is more complicated to use. All ufw commands need to be run as root or with sudo.

3.2.1 Setting up ufw

Important note before beginning: If you are configuring ufw over ssh (port 22) or some other remote managment program then make sure to allow that program through before enabling ufw or you will disconnect yourself and need physical access to the machine.

Run `ufw enable` to cause ufw to enforce its rules and `ufw disable` to turn off enforcement of its rules. List the rules with `ufw status`. The rules can be listed with numbers by running `ufw status numbered`. The order of rules is important because rules with lower numbers supercede rules with higher numbers. If your first rule is to allow all incoming and outgoing connections then all other rules have no effect and are useless. It is a good idea to allow outgoing connections by default.

3.2.2 Allowing and Denying Incoming Connections

Change the word 'allow' to 'deny' in any of the following commands to make them deny incoming connections instead of accept them.

- Run `ufw allow <port>` to allow incoming connections on <port>
- Run `ufw allow <port>/<protocol>` to allow incoming connections on port <port> where protocol is either tcp or udp.
- Run `ufw allow from <ip-address>` to allow incoming connections from <ip-address>.
- Run `ufw allow from <ip-address> to any port <port>` to allow <ip-address> to connect to <port>.
- Run `ufw allow from <ip-address> to any port <port> proto <protocol>` to allow <ip-address> to connect to <port> using <protocol>.

3.2.3 Ordering Rules

Rules in the list should start specific and become more general as the list progresses. To insert a rule run `ufw insert <number> <rule>` to insert <rule> above the rule currently at <number>. Example allow ssh when there are already some rules: `ufw insert 7 allow 22/tcp`. Delete a rule with `ufw delete <number>`. A typical setup usually has a list of "deny ip address" and then a list of "allow port".

3.2.4 Logging

Enable/disable logging with `ufw logging on` and `ufw logging off`. Logs are usually located at `/var/log/ufw.log[.X]`.

3.3 Setting up Public Key Authentication for an SSH Server

This section assumes that you already have a private key named id_rsa and a public key named id_rsa.pub.

3.3.1 Editing the SSH Server Config

Edit /etc/ssh/sshd_config as root. Change the line `PubkeyAuthentication no` to `PubkeyAuthentication yes`. Take note of the value for `AuthorizedKeysFile`. Password authentication can be disabled by changing the line `PasswordAuthentication yes` to `PasswordAuthentication no`.

3.3.2 Adding the Public Key to the Server

The file that stores the servers public keys is specified in /etc/ssh/sshd_config. Run `cat id_rsa.pub >> /path/to/authorized_keys` to add the public key id_rsa.pub to the list of authorized keys.

3.4 Netcat (nc)

Note that using a port below 1024 requires root or sudo. Also note that <ip-address> can be interchanged with a domain name unless the `-n` flag is specified.

3.4.1 Sending Data from a Server

Run `nc <ip-address> <port> < <file>` where <ip-address> is the listening clients ip address, <port> is the port number that the client is listening on and <file> is the data to be sent. An example looks like `nc 192.168.1.3 4040 < info.txt`. Run `<command-with-output> | nc <ip-address> <port>` to redirect a commands output (like echo or cat) to netcat to have it send that data. You may need to quit this with Ctrl-C.

3.4.2 Receiving Data as a Client

Run `nc -l <port> > <file>` to write the data received on <port> othe file <file>. An example looks like `nc 4040 > received.txt`. Run `nc -l <port> | <command>` to redirect the received data to a command. Warning: this can be dangerous especially if the command is bash.

3.4.3 Scanning Ports with Netcat

Run `nc -v -z -w1 <ip-address> <start-port>-<end-port>` to check if a port can be connected to. Remove the `-z` flag if you wish to send data (from a file or a stream).

3.5 HTTP Requests, Headers, and SSL

3.5.1 Sending and Receiving HTTP Requests with Headers via netcat

Run `nc <ip-address> <port>` to open a connection to a webserver on <port> at <ip-address>. Then enter the http request. The request could also be written to a file and then directed at netcat with `nc <ip-address> <port> < <filename>`. An example GET request is shown below:

```
GET /path/page.html HTTP/1.1
Host: <ip-address>
```

Then press enter twice and the webserver should return a http header followed by the a response to the http request. The example should return the html file located at /path/-page.html.

3.5.2 Sending and Receiving through SSL/TLS (HTTPS)

Run `openssl s_client -connect <ip-address>:<port>`. The command will output the SSL certificate and some other information. A http header can then be entered into the terminal. Press enter twice and the server should respond with a http header and appropriate response. The example above also works here.

3.6 Port Forwarding Through SSH Tunneling

Note that ports below 1024 require root or sudo.

3.6.1 Forward a Local Port to a Local Port

Run `ssh -L <alternate-port>:<website>:<original-port> <host>`, where <host> is the name of your computer, to locally tunnel traffic to <website> through localhost on <alternate-port>. <original-port> should be the port that <website> will respond on (usually 80).

3.6.2 Forward a Remote Port to a Local Port

In this scenario a remote machine refered to with <remote-ip> has a service running on <remote-port> but is only listening on its localhost. The following command makes the service available on the local machine on <local-port>: `sudo ssh -L localhost:<local-port>:localhost:<remote-port> -N <remote-ip>`.

3.6.3 Using SSH as a SOCKS Proxy

This command will tunnel all traffic on <local-port> through <remote-ip>: `ssh -N -D <local-port> <remote-ip>`. Then configure the local web browser to use a SOCKS4 proxy on localhost and <local-port>.

4. Installing Programs

4.1 Installing from Source

Most source code distributions come with a configure script, and a make file.

4.1.1 Configure

Usually run `./configure` to prepare to compile the software. This may say you need to install some libraries to compile the code.

4.1.2 Make

Make is a program that reads a ”make” file which is a list of commands and arguments to compile the source code and various other tasks. Before other tasks like ”install” or ”clean” the source code must be first compiled just by running `make`. You need to watch this progress because it will often fail early saying that a library is missing. If it fails you should run make again after installing the required package.

4.1.3 Install

make install

The make file should contain an ”install” section called by running `make install`(maybe as root) in the same directory as the configure script. This copies the compiled binaries to the user or system binary locations.

checkinstall

The program checkinstall creates a package from the compiled binaries that can be installed and managed through the system package manager. This makes removing programs easier because the alternative is to hope that the make file incules some type of remove or uninstall option or to reverse engineer it manually by deleting the files it creates.

4.2 Installing a .deb

These commands need to be run as root or with sudo. Run `dpkg -i /path/file.deb` to install a .deb file. Then run `apt-get install -f` to install dependencies. If you don't have apt and want to know the dependencies then run `dpkg -I /path/file.deb`.

5. Task Managment

5.1 Scheduling Tasks with cron

Note that different users and root have different crontabs. Run with sudo or su as a user to use other crontabs.

5.1.1 List Scheduled Tasks

Run `crontab -l` to list current scheduled tasks

5.1.2 Edit crontab

Run `crontab -e` to edit the crontab of the current user. Exiting and saving changes will cause them to take effect. The first five values on a line in a crontab indicate when it runs. The sixth value indicates the command to run. The command is a bash command and thus can be an executable program or script. Valid values for the first five arguments are numbers in the appropriate range or * to run on all valid numbers. The five values are often abbreviated as `m h dom mon dow` and are described as follows:

- **Minute(m)**: 0-59
- **Hour(h)**: 0-23 (0 is 12am midnight)
- **Day of Month(dom)**: 1-31
- **Month(mon)**: 1-12
- **Day of Week(dow)**: 0-6 (0 is Sunday)

5.1.3 Example crontab

```
#all days midnight to 5am
0 0 * * * /path/backup.sh start
0 5 * * * /path/backup.sh stop
#weekdays 9am-4pm
0 9 * * 1-5 /path/upload.sh start
0 16 * * 1-5 /path/upload.sh stop
#first of every month at midnight
```

```
0 0 1 * * /path/update.sh
```

5.2 Listing and Monitoring Programs

5.2.1 ps

Run `ps aux` to list all programs running on the system. The pid is often useful for communicating with the process. This is often combined with grep or find. An example looks like: `ps aux | grep -i firefox`

5.2.2 top

The program top and the more featured htop show the running processes in real time.

5.2.3 watch

Run `watch <command>` to show <command>'s output fullscreen and refresh it every two seconds. Run `watch -n <sec> <command>` to refresh every <sec> seconds.

5.2.4 pidof

Run `pidof <program-name>` to get the pid of <program-name>. This is useful for combining with other commands such as kill. Example using pidof: `kill $(pidof vi)`

5.2.5 iotop

Run `iotop` to show disk usage of different processes. This must be run as root or with sudo.

5.2.6 lsof

List Processes Using a File

Run `lsof <file>`

List Processes Using Files in a Directory

Non-recursively: `lsof +d <directory>`
Recursively: `lsof +D <directory>`

List Files Opened by a Specific User

Run `lsof -u <username>`

List Files Used by a Process ID

Run `lsof -p <pid>`

5.3 Terminating Programs

5.3.1 Using Signals with kill

Run `kill <pid>` to send the SIGTERM signal to <pid>. The SIGTERM signal tells the program to exit in a nice way. Run `kill -9 <pid>` to send the SIGKILL signal and terminate the program in a more forceful way. There are other signals that can be sent with kill that do not try to exit the program but communicate in other ways, these can be listed with `kill -l`.

5.3.2 Using top and htop

Pressing ”k” in top and htop allows for sending signals to programs.

6. User and Group Managment

6.1 Substitute User (su and sudo)

6.1.1 su

Substitute user (su) is used to login as another user and perform commands as them. Run `su <username>` to login as <username>. This command may require root or sudo. To run it with sudo it would be `sudo su <username>`.

6.1.2 sudo

Substitute user do is used to run a command as another user. Run `sudo <command>` to run <command> as root. Run `sudo -u <username> <command>` to run <command> as <user>.

6.1.3 visudo

The command `visudo` is used to change who can use sudo. It must be run as root or with sudo. A line that allows a user to use sudo looks like `<username> ALL=(ALL:ALL) ALL`. For a group the username would be a groupname and prefixed with a percent symobl (%).

6.2 Listing Users and Groups

6.2.1 Commands

With IDs

Run `id <username>` to list the user id, group id, and groups of that user.

Without IDs

Run `groups <username>` to list the names of the groups without IDs.

Password Information for a User

To show the information in /etc/shadow in a nice format and without the hash run `chage -l <username>`.

6.2.2 Files

Each line in the following files is a series of fields separated as by colons. The /etc/passwd and /etc/shadow files and /etc/group and /etc/gshadow files should be consistent in terms of users and groups listed.

/etc/passwd

The file /etc/passwd lists user accounts, IDs, and shells. The fields are as follows:

- **Username**
- **x**: Placeholder for password information.
- **User ID**
- **Group ID**
- **Comment**: Used to describe the user, this often contains commas for different contact information.
- **Home Directory**
- **User Shell**: /bin/bash or /bin/sh for users that can login. Often /bin/false or /bin/nologin for users that are not allowed to login.

Example /etc/passwd line:

```
wsmith:x:1000:1000:Will Smith,,,:/home/wsmith:/bin/bash
```

/etc/shadow

The file /etc/shadow is where hashes of passwords are stored. This file should only be readable by root and members of the shadow group and only writable by root. See file permissions below. The fields are as follows:

- **Username**
- **Password Hash(salted)**: An asterisk denotes that the user cannot login.
- **Last Password Change**: In unix time stamp format.
- **Days Until Password Change Permitted**: 0 indicates no restrictions.
- **Days Until Password Change Required**: 99999 indicates no expiration.
- **Days of Warning Before Password Change Required**
- **Days Until Account is Made Inactive**
- **When Account Expires**: In unix time stamp format.

- **Unused**

Example /etc/shadow line:
`root:notmyrealpasswordhash:16530:0:99999:7:::`

/etc/group

The file /etc/groups lists groups and IDs. The fields are as follows:

- **Group Name**
- **x**: Placeholder for password information
- **Group ID**
- **Group Members**: Comma separated list.

Example /etc/group line:
`adm:x:4:syslog,wsmith`

/etc/gshadow

The file /etc/gshadow lists groups and what users are members of the groups. The fields are as follows:

- **Group Name**
- **Hashed Group Password**: An asterisk indicates no password use available. An exclamation mark means the password is locked and no password use is available.
- **Group Administrators**: Comma separated list.
- **Group Members**: Comma separated list. Members can access the group without a password.

Example /etc/gshadow line:
`adm:*::syslog,wsmith`

6.3 Creating and Modifying Users and Groups

Note: Some commands may require root or sudo and some changes do not take effect until users logout and log back in.

6.3.1 Creating, Modifying, and Removing a User

Use the commands `adduser` and `deluser` most of the time. The commands `useradd`, `usermod`, and `userdel` are lower level utilities and are less ”friendly”.

Creating a User

Run `adduser <username>` to create a user with <username>. The user will not be able to login until they have a password. Run `passwd <username>` to set a password for them.

Adding the User to a Group

Run `adduser <username> <groupname>` to add the user to the specified group. Run `usermod -G <group-list> <username>` where <group-list> is a comma separated list of groups(no spaces) to add a user to multiple groups quickly.

Removing a User from a Group

Run `deluser <username> <groupname>` to remove <username> from <groupname>.

Removing a User

Run `deluser <username>` to remove <username>. Note that this does not remove the users home directory or files by default. Helpful flags include `--remove-home` and `--remove-all-files`.

6.3.2 Creating, Modifying, and Removing a Group

Creating a Group

Run `groupadd <groupname>` to create a group.

Renaming a Group

Run `groupmod -n <newname> <groupname>` to rename group <groupname> to <newname>.

Removing a Group

Run `groupdel <groupname>` to remove a group.

6.4 File Permissions, Ownership, and umask

6.4.1 File Permissions and Type

Run `ls -l` to show file permissions and ownership for files.

Type

The first field of the detailed list is the file permissions and file type. The first character is either a ”d” for directory or a ”-” for a file.

File Permissions

Following are three pairs of three characters. The pairs respectively are the owner, group, and other permissions. A ”-” character indicates that the option is not enabled.
For a file the ”r” indicates readability, ”w” writability, and ”x” the ability to execute.
For directories, ”r” offers the ability to list the contents of a directory, ”w” the ability to modify the list of the contents (rm, touch, mv, etc.) if the ”x” bit is also set, and ”x” offers the ability to cd into the directory.

Changing File Permissions with chmod

The ”rwx” pairs can be refered to by numbers where the binary equivalent of the number enables or disables the ”rwx”. For example 7 has the binary equivalent of 111 so it corresponds to ”rwx” while 5 has the binary equivalent of 101 so it corresponds to ”r-x”. Run `chmod <mode> <file>` where <mode> is three digits less than or equal to 7 that represent the owner, group, and other permissions.

To keep your home directory private run `chmod 700 <myhome>`. If you want another use to be able to access certain subdirectories and files (with other readable set) of your home directory but not be able to list the contents then run `chmod 701 <myhome>`. Files with passwords or sensitive data should always have no permissions available for the ”other” category.

Other File Permission Notation

File permissions can also be changed by first specifying user(u), group(g), and or other(o) and then +/- r, w, and or x. For example `chmod ug+x myscript.sh` allows the owner and group members to execute myscript.sh.

6.4.2 Ownership

Following the file permissions in a `ls -l` are the owner name and group name.

Changing Ownership

Run `chown <username> <file>` to change ownership of a file to <username> and run `chgrp <groupname <file>` to change group ownership fo a file to <groupname>.
Run `chown <username>:<groupname> <file>` to change the owner and group of a file.
Run `chown -R <username>:<groupname> <directory>` to recursively change the ownership for all subfiles and subfolders.

6.4.3 umask

Umask is used to control the file permissions for files and directories when they are created. The umask is the complement of the normal permissions and a binary 1 is used to remove permissions. For example, 0022 causes files to be created with 755 or ”rwxr-xr-x”. Umask has an extra digit at the beginning for special modes not described here. Run `umask` to show the current umask and run `umask <mode>` to set the umask to <mode>.

7. System Access Logs

Note: This is not a complete security reference. This is only a list of logs and commands for a few basic services on a system. This is not enough information to check a system doing anything more than a default installation.

7.1 Active Logins

Commands that show information about logged in users:

- `who`: Shows logged in users, what terminals, and when.
- `w`: Shows logged in users and what programs they are running.
- `users`: Shows only usernames of logged in users.

7.2 Login History

Commands and files that show login history:

- `last`: Shows when users logged in and out and if they are still logged in.
- `lastb`: Shows bad login attempts. Must be root or use sudo to run this.
- `lastlog`: Shows the last time that a user logged in.
- `/var/log/auth`: This file shows logins and attempts for the ssh server and contains ip addresses.

7.3 Application Logs

Application specific logs:

- `/var/log/ufw.log`: Log for uncomplicated firewall.
- `/var/log/vsftpd.log`: Log for vsftpd FTP server.
- `/var/log/apache2/access.log` and `/var/log/apache2/error.log`: Logs for the apache webserver.