Contents

Chapter 1: Getting started with GNU/Linux

Section 1.1: Useful shortcuts

Using The Terminal

> The examples in this document assume that you are using a POSIX-compliant (such as **bash**, **sh**, **zsh**, **ksh**) shell.

Large portions of GNU/Linux functionality are achieved using the terminal. Most distributions of Linux include terminal emulators that allow users to interact with a shell from their desktop environment. A shell is a command-line interpreter that executes user inputted commands. **Bash** (Bourne Again SHell) is a common default shell among many Linux distributions and is the default shell for macOS.

These shortcuts will work if you are using **Bash** with the *emacs* keybindings (set by default):

Open terminal

- `Ctrl + Alt + T` or `Super + T`

Cursor movement

- `Ctrl + A` Go to the beginning of the line you are currently typing on.
- `Ctrl + E` Go to the end of the line you are currently typing on.
- `Ctrl + XX` Move between the beginning of the line and the current position of the cursor.
- `Alt + F` Move cursor forward one word on the current line.
- `Alt + B` Move cursor backward one word on the current line.
- `Ctrl + F` Move cursor forward one character on the current line.
- `Ctrl + B` Move cursor backward one character on the current line.

Text manipulation

- `Ctrl + U` Cut the line from the current position to the beginning of the line, adding it to the clipboard. If you are at the end of the line, cut the entire line.
- `Ctrl + K` Cut the line from the current position to the end of the line, adding it to the clipboard. If you are at the beginning of the line, cut the entire line.
- `Ctrl + W` Delete the word before the cursor, adding it to the clipboard.
- `Ctrl + Y` Paste the last thing from the clipboard that you cut recently (undo the last delete at the **current** cursor position).
- `Alt + T` Swap the last two words before the cursor.
- `Alt + L` Make lowercase from cursor to end of word.
- `Alt + U` Make uppercase from cursor to end of word.
- `Alt + C` Capitalize to end of word starting at cursor (whole word if cursor is at the beginning of word).
- `Alt + D` Delete to end of word starting at cursor (whole word if cursor is at the beginning of word).
- `Alt + .` Prints the last word written in previous command.
- `Ctrl + T` Swap the last two characters before the cursor.

History access

- `Ctrl + R` Lets you search through previously used commands.
- `Ctrl + G` Leave history searching mode without running a command.
- `Ctrl + J` Lets you copy current matched command to command line without running it, allowing you to

make modifications before running the command.
- `Alt + R` Revert any changes to a command you've pulled from your history, if you've edited it.
- `Ctrl + P` Shows last executed command, i.e. walk back through the command history (Similar to up arrow).
- `Ctrl + N` Shows next executed command, i.e. walk forward through the command history (Similar to down arrow).

Terminal control

- `Ctrl + L` Clears the screen, similar to the clear command.
- `Ctrl + S` Stop all output to the screen. This is useful when running commands with lots of long output. But this doesn't stop the running command.
- `Ctrl + Q` Resume output to the screen after stopping it with Ctrl+S.
- `Ctrl + C` End currently running process and return the prompt.
- `Ctrl + D` Log out of the current shell session, similar to the exit or logout command. In some commands, acts as End of File signal to indicate that a file end has been reached.
- `Ctrl + Z` Suspends (pause) currently running foreground process, which returns shell prompt. You can then use `bg` command allowing that process to run in the background. To again bring that process to foreground, use `fg` command. To view all background processes, use `jobs` command.
- `Tab` Auto-complete files and directory names.
- `Tab Tab` Shows all possibilities, when typed characters doesn't uniquely match to a file or directory name.

Special characters

- `Ctrl + H` Same as Backspace.
- `Ctrl + J` Same as Return (historically Line Feed).
- `Ctrl + M` Same as Return (historically Carriage Return).
- `Ctrl + I` Same as Tab.
- `Ctrl + G` Bell Character.
- `Ctrl + @` Null Character.
- `Esc` Deadkey equivalent to the `Alt` modifier.

Close Terminal

- `Ctrl + Shift + W` To close terminal tab.
- `Ctrl + Shift + Q` To close entire terminal.

Alternatively, you can switch to the *vi* keybindings in **bash** using `set -o vi`. Use `set -o emacs` to switch back to the *emacs* keybindings.

Section 1.2: File Management Commands

Linux uses some conventions for present and parent directories. This can be a little confusing for beginners.

Whenever you are in a terminal in Linux, you will be in what is called the *current working directory*. Often your command prompt will display either the full working directory, or just the last part of that directory. Your prompt could look like one of the following:

```
user@host ~/somedir $
user@host somedir $
user@host /home/user/somedir $
```

which says that your current working directory is `/home/user/somedir`.

In Linux `..` represents the parent directory and `.` represents the current directory.

Therefore, if the current directory is `/home/user/somedir`, then `cd ../somedir` will not change the working directory.

The table below lists some of the most used file management commands

Directory navigation

Command	Utility
`pwd`	Get the full path of the current working directory.
`cd -`	Navigate to the last directory you were working in.
`cd ~` or just `cd`	Navigate to the current user's home directory.
`cd ..`	Go to the parent directory of current directory (mind the space between `cd` and `..`)

Listing files inside a directory

Command	Utility
`ls -l`	List the files and directories in the current directory in long (table) format (It is recommended to use -l with ls for better readability).
`ls -ld dir-name`	List information about the directory `dir-name` instead of its contents.
`ls -a`	List all the files including the hidden ones (File names starting with a `.` are hidden files in Linux).
`ls -F`	Appends a symbol at the end of a file name to indicate its type (`*` means executable, `/` means directory, `@` means symbolic link, `=` means socket, `|` means named pipe, `>` means door).
`ls -lt`	List the files sorted by last modified time with most recently modified files showing at the top (remember -l option provides the long format which has better readability).
`ls -lh`	List the file sizes in human readable format.
`ls -lR`	Shows all subdirectories recursively.
`tree`	Will generate a tree representation of the file system starting from the current directory.

File/directory create, copy and remove

Command	Utility
`cp -p source destination`	Will copy the file from `source` to *destination*. -p stands for preservation. It preserves the original attributes of file while copying like file owner, timestamp, group, permissions etc.
`cp -R source_dir destination_dir`	Will copy source directory to specified destination recursively.
`mv file1 file2`	In Linux there is no **rename** command as such. Hence `mv` moves/renames the file1 to file2.
`rm -i filename`	Asks you before every file removal for confirmation. **IF YOU ARE A NEW USER TO LINUX COMMAND LINE, YOU SHOULD ALWAYS USE `rm -i`.** You can specify multiple files.
`rm -R dir-name`	Will remove the directory `dir-name` recursively.
`rm -rf dir-name`	Will remove the directory `dir` recursively, ignoring non-existent files and will never prompt for anything. **BE CAREFUL USING THIS COMMAND!** You can specify multiple directories.
`rmdir dir-name`	Will remove the directory `dir-name`, if it's empty. This command can only remove empty directories.
`mkdir dir-name`	Create a directory `dir-name`.
`mkdir -p dir-name/dir-name`	Create a directory hierarchy. Create parent directories as needed, if they don't exist. You can specify multiple directories.
`touch filename`	Create a file `filename`, if it doesn't exist, otherwise change the timestamp of the file to current time.

File/directory permissions and groups

Command	Utility

`chmod <specification> filename`	Change the file permissions. Specifications = `u` user, `g` group, `o` other, `+` add permission, `-` remove, `r` read, `w` write,`x` execute.
`chmod -R <specification> dir-name`	Change the permissions of a directory recursively. To change permission of a directory and everything within that directory, use this command.
`chmod go=+r myfile`	Add read permission for the owner and the group.
`chmod a +rwx myfile`	Allow all users to read, write or execute `myfile`.
`chmod go -r myfile`	Remove read permission from the group and others.
`chown owner1 filename`	Change ownership of a file to user `owner1`.
`chgrp grp_owner filename`	Change primary group ownership of file `filename` to group `grp_owner`.
`chgrp -R grp_owner dir-name`	Change primary group ownership of directory `dir-name` to group `grp_owner` recursively. To change group ownership of a directory and everything within that directory, use this command.

Section 1.3: Hello World

Type the following code into your terminal, then press `Enter`:

```
echo "Hello World"
```

This will produce the following output:

```
Hello World
```

Section 1.4: Basic Linux Utilities

Linux has a command for almost any tasks and most of them are intuitive and easily interpreted.

Getting Help in Linux

Command	Usability
`man <name>`	Read the manual page of <name>.
`man <section> <name>`	Read the manual page of <name>, related to the given section.
`man -k <editor>`	Output all the software whose man pages contain <editor> keyword.
`man -K <keyword>`	Outputs all man pages containing <keyword> within them.
`apropos <editor>`	Output all the applications whose one line description matches the word *editor*. When **not able to recall** the name of the application, use this command.
`help`	In Bash shell, this will display the list of all available bash commands.
`help <name>`	In Bash shell, this will display the info about the <name> bash command.
`info <name>`	View all the information about <name>.
`dpkg -l`	Output a list of all installed packages on a Debian-based system.
`dpkg -L packageName`	Will list out the files installed and path details for a given package on Debian.
`dpkg -l | grep -i <edit>`	Return all .deb installed packages with <edit> irrespective of cases.
`less /var/lib/dpkg/available`	Return descriptions of all available packages.
`whatis vim`	List a one-line description of vim.
`<command-name> --help`	Display usage information about the <tool-name>. Sometimes `command -h` also works, but not for all commands.

User identification and who is who in Linux world

Command	Usability
`hostname`	Display hostname of the system.

`hostname -f`	Displays Fully Qualified Domain Name (FQDN) of the system.
`passwd`	Change password of current user.
`whoami`	Username of the users logged in at the terminal.
`who`	List of all the users currently logged in as a user.
`w`	Display current system status, time, duration, list of users currently logged in on system and other user information.
`last`	Who recently used the system.
`last root`	When was the last time **root** logged in as user.
`lastb`	Shows all bad login attempts into the system.
`chmod`	Changing permissions - read,write,execute of a file or directory.

Process related information

Command	Usability
`top`	List all processes sorted by their current system resource usage. Displays a continually updated display of processes (By default 3 seconds). Use `q` key to exit top.
`ps`	List processes currently running on current shell session
`ps -u root`	List all of the processes and commands root is running
`ps aux`	List all the processes by all users on the current system

Section 1.5: Searching for files by patterns in name/contents

A common and task of someone using the Linux Command Line (shell) is to search for files/directories with a certain name or containing certain text. There are 2 commands you should familiarise yourself with in order to accomplish this:

Find files by name

```
find /var/www -name '*.css'
```

This will print out the full path/filename to all files under `/var/www` that end in `.css`. Example output:

```
/var/www/html/text-cursor.css
/var/www/html/style.css
```

For more info:

```
man find
```

Find files containing text

```
grep font /var/www/html/style.css
```

This will print all lines containing the pattern `font` in the specified file. Example output:

```
font-weight: bold;
font-family: monospace;
```

Another example:

```
grep font /var/www/html/
```

This doesn't work as you'd hoped. You get:

```
grep: /var/www/html/: Is a directory
```

You need to `grep` recursively to make it work, using the `-R` option:

```
grep -R font /var/www/html/
```

Hey nice! Check out the output of this one:

```
/var/www/html/admin/index.php:  echo '<font color=red><b>Error: no dice</b></font><br/>';
/var/www/html/admin/index.php:  echo '<font color=red><b>Error: try again</b></font><br/>';
/var/www/html/style.css:  font-weight: bold;
/var/www/html/style.css:  font-family: monospace;
```

Notice that when `grep` is matching multiple files, it prefixes the matched lines with the filenames. You can use the `-h` option to get rid of that, if you want.

For more info:

```
man grep
```

Section 1.6: File Manipulation

Files and directories (another name for folders) are at the heart of Linux, so being able to create, view, move, and delete them from the command line is very important and quite powerful. These file manipulation commands allow you to perform the same tasks that a graphical file explorer would perform.

Create an empty text file called `myFile`:

```
touch myFile
```

Rename `myFile` to `myFirstFile`:

```
mv myFile myFirstFile
```

View the contents of a file:

```
cat myFirstFile
```

View the content of a file with pager (one screenful at a time):

```
less myFirstFile
```

View the first several lines of a file:

```
head myFirstFile
```

View the last several lines of a file:

```
tail myFirstFile
```

Edit a file:

```
vi myFirstFile
```

See what files are in your current working directory:

```
ls
```

Create an empty directory called `myFirstDirectory`:

```
mkdir myFirstDirectory
```

Create multi path directory: (creates two directories, src and myFirstDirectory)

```
mkdir -p src/myFirstDirectory
```

Move the file into the directory:

```
mv myFirstFile myFirstDirectory/
```

You can also rename the file:

```
user@linux-computer:~$ mv myFirstFile secondFileName
```

Change the current working directory to `myFirstDirectory`:

```
cd myFirstDirectory
```

Delete a file:

```
rm myFirstFile
```

Move into the parent directory (which is represented as `..`):

```
cd ..
```

Delete an empty directory:

```
rmdir myFirstDirectory
```

Delete a non-empty directory (i.e. contains files and/or other directories):

```
rm -rf myFirstDirectory
```

Make note that when deleting directories, that you delete `./` not `/` that will wipe your whole filesystem.

Section 1.7: File/Directory details

The `ls` command has several options that can be used together to show more information.

Details/Rights

The `l` option shows the file permissions, size, and last modified date. So if the root directory contained a dir called `test` and a file `someFile` the command:

```
user@linux-computer:~$ ls -l
```

Would output something like

```
-rw-r--r-- 1 user users   70 Jul 22 13:36 someFile.txt
drwxrwxrwx 2 user users 4096 Jul 21 07:18 test
```

The permissions are in format of `drwxrwxrwx`. The first character represents the file type `d` if it's a directory `-` otherwise. The next three `rwx` are the permissions the user has over the file, the next three are the permissions the group has over the file, and the last three are the permissions everyone else has over the file.

The `r` of `rwx` stands for if a file can be read, the `w` represents if the file can be modified, and the `x` stands for if the file can be executed. If any permission isn't granted a `-` will be in place of `r`, `w`, or `x`.

So from above `user` can read and modify `someFile.txt` but the group has only read-only rights.

To change rights you can use the `chmod ### fileName` command if you have sudo rights. `r` is represented by a value of 4, `w` is represented by 2, and `x` is represented by a 1. So if only you want to be able to modify the contents to the `test` directory

```
Owner rwx = 4+2+1 = 7
Group r-x = 4+0+1 = 5
Other r-x = 4+0+1 = 5
```

So the whole command is

```
chmod 755 test
```

Now doing a `ls -l` would show something like

```
drwxr-xr-x 2 user users 4096 Jul 21 07:20 test
```

Readable Size

Used in conjunction with the `l` option the `h` option shows file sizes that are human readable. Running

```
user@linux-computer:~$ ls -lh
```

Would output:

```
total 4166
-rw-r--r-- 1 user users   70 Jul 22 13:36 someFile.txt
drwxrwxrwx 2 user users 4.0K Jul 21 07:18 test
```

Hidden

To view hidden files use the `a` option. For example

```
user@linux-computer:~$ ls -a
```

Might list

```
.profile
someFile.txt
test
```

Total Directory Size

To view the size of the current directory use the `s` option (the `h` option can also be used to make the size more readable).

```
user@linux-computer:~$ ls -s
```

Outputs

```
total 4166
someFile.txt      test
```

Recursive View

Lets say `test` directory had a file `anotherFile` and you wanted to see it from the root folder, you could use the `R` option which would list the recursive tree.

```
user@linux-computer:~$ ls -R
```

Outputs

```
.:
someFile.txt    test

./test:
anotherFile
```

Chapter 2: Detecting Linux distribution name and version

Section 2.1: Detect what debian-based distribution you are working in

Just execute `lsb_release -a`.

On Debian:

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux testing (stretch)
Release:        testing
Codename:       stretch
```

On Ubuntu:

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.4 LTS
Release:        14.04
Codename:       trusty
```

In case when you don't have `lsb_release` installed you may want to try some guessing, for example, there is a file `/etc/issue` that often contains distribution name. For example, on ubuntu:

```
$ cat /etc/issue
Ubuntu 12.04.5 LTS \n \l
```

Don't use file `/etc/debian_version` because its contents do not match distribution name!

Note that this will also work on non-Debian-family distributions like Fedora, RHEL, or openSUSE — but that `lsb_release` may not be installed.

Section 2.2: Detect what systemd-based distribution you are using

This method will work on modern versions of Arch, CentOS, CoreOS, Debian, Fedora, Mageia, openSUSE, Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Ubuntu, and others. This wide applicability makes it an ideal as a first approach, with fallback to other methods if you need to also identify older systems.

Look at `/etc/os-release`. In specific, look at variables `NAME`, `VERSION`, `ID`, `VERSION_ID`, and `PRETTY_NAME`.

On Fedora, this file might look like:

```
NAME=Fedora
VERSION="24 (Workstation Edition)"
ID=fedora
VERSION_ID=24
PRETTY_NAME="Fedora 24 (Workstation Edition)"
ANSI_COLOR="0;34"
```

```
CPE_NAME="cpe:/o:fedoraproject:fedora:24"
HOME_URL="https://fedoraproject.org/"
BUG_REPORT_URL="https://bugzilla.redhat.com/"
REDHAT_BUGZILLA_PRODUCT="Fedora"
REDHAT_BUGZILLA_PRODUCT_VERSION=24
REDHAT_SUPPORT_PRODUCT="Fedora"
REDHAT_SUPPORT_PRODUCT_VERSION=24
PRIVACY_POLICY_URL=https://fedoraproject.org/wiki/Legal:PrivacyPolicy
VARIANT="Workstation Edition"
VARIANT_ID=workstation
```

On CentOS, this file might look like this:

```
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"
```

This file is documented on the freedesktop web site; in principle, it is not systemd specific — but it will exist on all systemd-based distributions.

From the bash shell, one can source the `/etc/os-release` file and then use the various variables directly, like this:

```
$ ( source /etc/os-release && echo "$PRETTY_NAME" )
Fedora 24 (Workstation Edition)
```

Section 2.3: Detect what RHEL / CentOS / Fedora distribution you are working in

Look at the contents of `/etc/redhat-release`

```
cat /etc/redhat-release
```

Here is the output from a Fedora 24 machine: `Fedora release 24 (Twenty Four)`

As mentioned in the debian-based response, you can also use the `lsb_release -a` command, which outputs this from a Fedora 24 machine:

```
LSB Version:    :core-4.1-amd64:core-4.1-noarch:cxx-4.1-amd64:cxx-4.1-noarch:desktop-4.1-
amd64:desktop-4.1-noarch:languages-4.1-amd64:languages-4.1-noarch:printing-4.1-amd64:printing-4.1-
noarch
Distributor ID:    Fedora
Description:    Fedora release 24 (Twenty Four)
Release:    24
Codename:    TwentyFour
```

Section 2.4: Uname - Print information about the current system

Uname is the short name for **u**nix **name**. Just type `uname` in console to get information about your operating system.

```
uname [OPTION]
```

If no *OPTION* is specified, `uname` assumes the `-s` option.

`-a` or `--all` - Prints all information, omitting `-p` and `-i` if the information is unknown.

Example:

```
> uname -a

SunOS hope 5.7 Generic_106541-08 sun4m sparc SUNW,SPARCstation-10
```

All the options:

-s, **--kernel-name**	Print the kernel name.
-n, **--nodename**	Print the network node hostname.
-r, **--kernel-release**	Print the kernel release.
-v, **--kernel-version**	Print the kernel version.
-m, **--machine**	Print the machine hardware name.
-p, **--processor**	Print the processor type, or "**unknown**".
-i, **--hardware-platform**	Print the hardware platform, or "**unknown**".
-o, **--operating-system**	Print the operating system.
--help	Display a help message, and exit.
--version	Display version information, and exit.

Section 2.5: Detect basic information about your distro

just execute `uname -a`.

On Arch:

```
$ uname -a
Linux nokia 4.6.4-1-ARCH #1 SMP PREEMPT Mon Jul 11 19:12:32 CEST 2016 x86_64       GNU/Linuxenter
code here
```

Section 2.6: Using GNU coreutils

So the GNU coreutils should be avaialable on all linux based systems (please correct me if I am wrong here).

If you do not know what system you are using you may not be able to directly jump to one of the examples above, hence this may be your first port of call.

```
$ uname -a
```

On my system this gives me the following...

```
Linux Scibearspace 3.16.0-4-amd64 #1 SMP Debian 3.16.7-ckt25-2+deb8u3 (2016-07-02) x86_64
```

```
GNU/Linux
```

Here you can see the following :

Scibearspace : the name of my pc

- Scibearspace : the name of my pc
- 3.16.0-4-amd64 : the kernel and architecture
- SMP Debian 3.16.7-CKT25-2+deb8u3 : tells me I am running debian with the 3.16 kernel
- Finaly the last part I am running debian 8 (update 3).

I would welcome any others to add in results for RHEL, and SuSe systems.

Section 2.7: Find your linux os (both debian & rpm) name and release number

Most of linux distros stores its version info in the /etc/lsb-release (debian) or /etc/redhat-release (RPM based) file. Using below generic command should get you past most of the Debian and RPM derivatives as Linux Mint and Cent-Os.

Example on Ubuntu Machine:

```
cat /etc/*release
```

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04 LTS"
```

Chapter 3: Getting information on a running Linux kernel

Section 3.1: Getting details of Linux kernel

We can use command uname with various options to get complete details of running kernel.

```
uname -a
```

```
Linux df1-ws-5084 4.4.0-64-generic #85-Ubuntu SMP Mon Feb 20 11:50:30 UTC 2017 x86_64
x86_64 x86_64 GNU/Linux
```

As per man page here few more options

Usage: uname [OPTION]...
Print certain system information. With no OPTION, same as -s.

```
-a, --all                print all information, in the following order,
except omit -p and -i if unknown:
-s, --kernel-name        print the kernel name
-n, --nodename           print the network node hostname
-r, --kernel-release     print the kernel release
-v, --kernel-version     print the kernel version
-m, --machine            print the machine hardware name
-p, --processor          print the processor type (non-portable)
-i, --hardware-platform  print the hardware platform (non-portable)
-o, --operating-system   print the operating system
--help     display this help and exit
--version  output version information and exit
```

Chapter 4: Shell

The shell executes a program in response to its prompt. When you give a command, the shell searches for the program, and then executes it. For example, when you give the command ls, the shell searches for the utility/program named ls, and then runs it in the shell. The arguments and the options that you provide with the utilities can impact the result that you get. The shell is also known as a CLI, or command line interface.

Section 4.1: Changing default shell

Most modern distributions will come with BASH (**B**ourne **A**gain **SH**ell) pre-installed and configured as a default shell.

The command (actually an executable binary, an ELF) that is responsible for changing shells in Linux is `chsh` (**ch**ange **sh**ell).

We can first check which shells are already installed and configured on our machine by using the `chsh -l` command, which will output a result similar to this:

```
[user@localhost ~]$ chsh -l
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
/usr/bin/fish
```

In some Linux distributions, `chsh -l` is invalid. In this case, the list of all available shells can be found at /etc/shells file. You can show the file contents with `cat`:

```
[user@localhost ~]$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
/usr/bin/fish
```

Now we can choose our new default shell, e.g. `fish`, and configure it by using `chsh -s`,

```
[user@localhost ~]$ chsh -s /usr/bin/fish
Changing shell for user.
Password:
Shell changed.
```

Now all that is left to do is preform a logoff-logon cycle, and enjoy our new default shell.

If you wish to change the default shell for a different user, and you have administrative privileges on the machine, you'll be able to accomplish this by using `chsh` as `root`. So assuming we want to change `user_2`'s default shell to fish, we will use the same command as before, but with the addition of the other user's username, `chsh -s /usr/bin/fish user_2`.

In order to check what the current default shell is, we can view the `$SHELL` environment variable, which points to the path to our default shell, so after our change, we would expect to get a result similar to this,

```
 ~ ↱ echo $SHELL
/usr/bin/fish
```

`chsh` options:

`-s shell`

Sets shell as the login shell.

`-l`, `--list-shells`

Print the list of shells listed in /etc/shells and exit.

`-h`, `--help`

Print a usage message and exit.

`-v`, `--version`

Print version information and exit.

Section 4.2: Basic Shell Utilities

Customizing the Shell prompt

Default command prompt can be changed to look different and short. In case the current directory is long default command prompt becomes too large. Using `PS1` becomes useful in these cases. A short and customized command pretty and elegant. In the table below `PS1` has been used with a number of arguments to show different forms of shell prompts. Default command prompt looks something like this: `user@host ~ $` in my case it looks like this: `bruce@gotham ~ $`. It can changed as per the table below:

Command	Utility
PS1='\w $ '	`~ $` shell prompt as directory name. In this case root directory is Root.
PS1='\h $ '	`gotham $` shell prompt as hostname
PS1='\u $ '	`bruce $` shell prompt as username
PS1='\t $ '	`22:37:31 $` shell prompt in 24 hour format
PS1='@ $ '	`10:37 PM` shell prompt in 12 hour time format
PS1='! $ '	`732` will show the history number of command in place of shell prompt
PS1='dude $ '	`dude $` will show the shell prompt the way you like

Some basic shell commands

Command	Utility
`Ctrl-k`	cut/kill
`Ctrl-y`	yank/paste
`Ctrl-a`	will take cursor to the start of the line
`Ctrl-e`	will take cursor to the end of the line
`Ctrl-d`	will delete the character after/at the cursor
`Ctrl-l`	will clear the screen/terminal
`Ctrl-u`	will clear everything between prompt and the cursor
`Ctrl-_`	will undo the last thing typed on the command line
`Ctrl-c`	will interrupt/stop the job/process running in the foreground

`Ctrl-r`	reverse search in history
`~/.bash_history`	stores last 500 commands/events used on the shell
`history`	will show the command history
`history | grep <key-word>`	will show all the commands in history having keyword <key-word> (useful in cases when you remember part of the command used in the past)

Section 4.3: Create Your Own Command Alias

If you are tired of using long commands in bash you can create your own command alias.

The best way to do this is to modify (or create if it does not exist) a file called .bash_aliases in your home folder. The general syntax is:

```
alias command_alias='actual_command'
```

where `actual_command` is the command you are renaming and `command_alias` is the new name you have given it. For example

```
alias install='sudo apt-get -y install'
```

maps the new command alias `install` to the actual command `sudo apt-get -y install`. This means that when you use **install** in a terminal this is interpreted by bash as **sudo apt-get -y install**.

Section 4.4: Locate a file on your system

Using bash you can easily locate a file with the `locate` command. For example say you are looking for the file mykey.pem:

```
locate mykey.pem
```

Sometimes files have strange names for example you might have a file like `random7897_mykey_0fidw.pem`. Let's say you're looking for this file but you only remember the mykey and pem parts. You could combine the `locate` command with `grep` using a pipe like this:

```
locate pem | grep mykey
```

Which would bring up all results which contain both of these pieces.

Note that not all systems have the `locate` utility installed, and many that do have not enabled it. `locate` is fast and efficient because it periodically scans your system and caches the names and locations for every file on it, but if that data collection is not enabled then it cannot tell you anything. You can use `updatedb` to manually initiate the filesystem scan in order to update the cached info about files on your filesystem.

Should you not have a working `locate`, you can fall back on the `find` utility:

```
find / -name mykey.pem -print
```

is roughly equivalent to `locate mykey.pem` but has to scan your filesystem(s) each time you run it for the file in question, rather than using cached data. This is obviously slower and less efficient, but more real-time. The `find` utility can do much more than find files, but a full description of its capabilities is beyond the scope of this example.

Chapter 5: Check Disk Space

Section 5.1: Investigate Directories For Disk Usage

Sometimes it may be required to find out which directory consuming how much disk space especially when you are used `df -h` and realized your available disk space is low.

du:

`du` command summarizes disk usage of the set of FILEs, recursively for directories.

It's often uses with `-sh` option:

```
-s, --summarize
display only a total for each argument
-h, --human-readable
print sizes in human readable format (e.g., 1K 234M 2G)
```

For summarizing disk usages of the files in the current directory we use:

```
du -sh *
```

Example output:

```
572K    Documents
208M    Downloads
4,0K    Music
724K    Pictures
4,0K    Public
4,0K    Templates
4,0K    Videos
```

We can also include hidden files with using:

```
du -sh .[!.]* *
```

Example output:

```
6,3M    .atom
4,0K    .bash_history
4,0K    .bash_logout
8,0K    .bashrc
350M    .cache
195M    .config
12K    .dbus
4,0K    .dmrc
44K    .gconf
60K    .gem
520K    .gimp-2.8
28K    .gnome
4,0K    .ICEauthority
8,3M    .local
8,0K    .nano
404K    .nv
36K    .pki
```

```
4,0K    .profile
8,0K    .ssh
0    .sudo_as_admin_successful
4,0K    .Xauthority
4,0K    .xsession-errors
4,0K    .xsession-errors.old
572K    Documents
208M    Downloads
4,0K    Music
724K    Pictures
4,0K    Public
4,0K    Templates
4,0K    Videos
```

Thirdly, you can add total to the output by adding ,-c, option:

```
du -sch .[!.]* *
```

Result:

```
.
.
.
4,0K    Templates
4,0K    Videos
769M    total
```

Most importantly using `du` command properly on the root directory is a life saving action to find out what application/service or user is consuming your disk space wildly. For example, in case of a ridiculously low level of disk space availability for a web and mail server, the reason could be a spam attack to your mail service and you can diagnose it just by using `du` command.

Investigate root directory for disk usage:

```
sudo du -sch /.[!.]* /*
```

Example output:

```
16K    /.VolumeIcon.icns
24K    /.VolumeIcon.png
13M    /bin
57M    /boot
4,0K    /cdrom
620K    /dev
13M    /etc
779M    /home
0    /initrd.img
406M    /lib
3,9M    /lib32
4,0K    /lib64
16K    /lost+found
4,0K    /media
4,0K    /mnt
367M    /opt
du: cannot access '/proc/18221/task/18221/fd/4': No such file or directory
du: cannot access '/proc/18221/task/18221/fdinfo/4': No such file or directory
du: cannot access '/proc/18221/fd/4': No such file or directory
```

```
du: cannot access '/proc/18221/fdinfo/4': No such file or directory
0    /proc
20K    /root
du: cannot access '/run/user/1000/gvfs': Permission denied
9,4M    /run
13M    /sbin
4,0K    /srv
0    /sys
72K    /tmp
3,5G    /usr
639M    /var
0    /vmlinuz
5,8G    total
```

Lastly, the best method forms when you add a threshold size value for directories to ignore small ones. This command will only show folders with more than 1GB in size which located under root directory up to the farthermost branch of the whole directory tree in your file system:

```
sudo du --threshold=1G -ch /.[!.]* /*
```

Example output:

```
1,4G    /usr/lib
1,8G    /usr/share
3,5G    /usr
5,8G    total
```

Section 5.2: Checking Disk Space

It's quite common to want to check the status of the various partitions/drives on your server/computer to see how full they are. The following command is the one you'll want to run:

```
df -h
```

This will produce output similar to the following:

```
[root@mail ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root
                       19G  1.6G   16G   9% /
tmpfs                 245M     0  245M   0% /dev/shm
/dev/sda1             485M   47M  413M  11% /boot
```

In this basic example, we can see that the `/` partition only has `9%` used.

For a more complex example that also covers using `df` to see various mountpoints, see below:

```
[root@mail ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VG-root   1.9T  1.7T   89G  95% /
/dev/mapper/VG-var    431G  145G  264G  36% /var
devtmpfs              7.8G  204K  7.8G   1% /dev
tmpfs                 7.8G  4.0K  7.8G   1% /dev/shm
/dev/md1              495M  126M  344M  27% /boot
ku.example.com:9421   2.5T  487G  2.0T  20% /mnt/test
tmpfs                 500M   86M  415M  18% /var/ngx_pagespeed_cache
```

In this example, we have a `/` partition that's `95%` full along with an additional `/var` partition that's only `36%` full.

It's got an external network mount of `2T` that's mounted on `/mnt/test` and a ramdisk/tmpfs mount of 500M mounted on `/var/ngx_pagespeed_cache`.

Chapter 6: Getting System Information

Collection of commands to fetch system related information.

Section 6.1: Statistics about CPU, Memory, Network and Disk (I/O operations)

To get general statistics about main components of Linux family of **stat** commands are extremely useful

CPU

To get processors related statistics you can use `mpstat` command but with some options it will provide better visibility:

```
$ mpstat 2 10
```

Memory

We all know command **free** to show amount of (remaining) RAM but to see all statistic including I/O operations:

```
$ vmstat 2 10
```

Disk

To get general information about your disk operations in real time you can utilise `iostat`.

```
$ iostat -kx 2
```

Network

To be able to see what is happening with your network services you can use **netstat**

```
$ netstat -ntlp # open TCP sockets
$ netstat -nulp # open UDP sockets
$ netstat -nxlp # open Unix sockets
```

But you can find useful monitoring to see network traffic in real time:

```
$ sudo iftop
```

Optional

To generate statistics in real time related to I/O operations across all components you can use `dstat`. That tool that is a versatile replacement for **vmstat**, `iostat` and `ifstat`

Section 6.2: Using tools like lscpu and lshw

By using tools like lscpu as lscpu is an easy way to get CPU information.

```
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
```

```
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 23
Stepping:              10
CPU MHz:               1998.000
BogoMIPS:              5303.14
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              2048K
NUMA node0 CPU(s):     0-3
```

By using tool lshw

```
$ lshw | grep cpu

df1-ws-5084
    description: Computer
    width: 64 bits
    capabilities: vsyscall32
  *-core
       description: Motherboard
       physical id: 0
     *-memory
          description: System memory
          physical id: 0
          size: 5881MiB
     *-cpu
          product: Intel(R) Pentium(R) CPU G3220 @ 3.00GHz
          vendor: Intel Corp.
          physical id: 1
          bus info: cpu@0
          size: 3GHz
          capacity: 3GHz
          width: 64 bits
```

Section 6.3: List Hardware

Ubuntu:

lshw is a small tool to extract detailed information on the hardware configuration of the machine. It can report exact memory configuration, firmware version, mainboard configuration, CPU version and speed, cache configuration, bus speed, etc.

```
$ sudo lshw | less (or more)
$ sudo lshw -html > myhardware.html
$ sudo lshw -xml > myhardware.xml
```

To show PCI info

```
$ lspci -tv
```

To see USB info

```
$ lsusb -tv
```

To display BIOS information

```
$ dmidecode -q | less
```

To see specific information about disk (disk sda in example) you can use:

```
$ hdparm -i /dev/sda
```

Few additional utilities/commands will help gather some extra information:

```
$ smartctl -A /dev/sda | grep Power_On_Hours # How long has this disk (system) been powered on in total
$ hdparm -tT /dev/sda # Do a read speed test on disk sda
$ badblocks -s /dev/sda # Test for unreadable blocks on disk sda
```

Section 6.4: Find CPU model/speed information

Ubuntu:

```
$ cat /proc/cpuinfo
```

Sample Output:

```
processor    : 0
vendor_id    : GenuineIntel
cpu family    : 6
model        : 15
model name    : Intel(R) Core(TM)2 Quad CPU    Q6600  @ 2.40GHz
stepping    : 11
cpu MHz        : 1596.000
cache size    : 4096 KB
physical id    : 0
siblings    : 4
core id        : 0
cpu cores    : 4
apicid        : 0
initial apicid    : 0
fpu        : yes
fpu_exception    : yes
cpuid level    : 10
wp        : yes
flags        : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts
acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant_tsc arch_perfmon pebs bts rep_good pni
dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm lahf_lm tpr_shadow vnmi flexpriority
bogomips    : 4800.18
clflush size    : 64
cache_alignment    : 64
address sizes    : 36 bits physical, 48 bits virtual
power management:
....
..
processor    : 3
vendor_id    : GenuineIntel
cpu family    : 6
```

```
model        : 15
model name    : Intel(R) Core(TM)2 Quad CPU    Q6600  @ 2.40GHz
stepping    : 11
cpu MHz        : 1596.000
cache size    : 4096 KB
physical id    : 0
siblings    : 4
core id        : 3
cpu cores    : 4
apicid        : 3
initial apicid    : 3
fpu        : yes
fpu_exception    : yes
cpuid level    : 10
wp        : yes
flags        : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts
acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant_tsc arch_perfmon pebs bts rep_good pni
dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm lahf_lm tpr_shadow vnmi flexpriority
bogomips    : 4800.30
clflush size    : 64
cache_alignment    : 64
address sizes    : 36 bits physical, 48 bits virtual
power management:
```

count processor (including cores):

```
$ grep -c processor /proc/cpuinfo
```

Section 6.5: Process monitoring and information gathering

Overall you have two ways to monitor processes at linux host

Static monitoring

Most widely used command is `ps` (i.e., process status) command is used to provide information about the currently running processes, including their process identification numbers (PIDs).

Here few useful options to gather specific information.

List processes in a hierarchy

```
$ ps -e -o pid,args --forest
```

List processes sorted by % cpu usage

```
$ ps -e -o pcpu,cpu,nice,state,cputime,args --sort pcpu | sed '/^ 0.0 /d'
```

List processes sorted by mem (KB) usage.

```
$ ps -e -orss=,args= | sort -b -k1,1n | pr -TW$COLUMNS
```

List all threads for a particular process ("firefox-bin" process in example)

```
$ ps -C firefox-bin -L -o pid,tid,pcpu,state
```

After finding specific process you can gather information related to it using **`lsof`** to list paths that process id has open

```
$ lsof -p $$
```

Or based on path find out list processes that have specified path open

```
$ lsof ~
```

Interactive monitoring

Most commonly known tool for dynamic monitoring is:

```
$ top
```

That mostly default command that have huge amount options to filter and represent information in real time (in comparison to `ps` command.

Still there are more advance options that can be considered and installed as `top` replacement

```
$ htop -d 5
```

or

```
$ atop
```

Which has ability to log all the activities into log file (default atop will log all the activity on every 600 seconds) To this list there are few specialised commands as `iotop` or `iftop`

```
$ sudo iotop
```

Chapter 7: ls command

Section 7.1: Options for ls command

Full list of options:

`ls -a` list all files including hidden file starting with '.'

`ls --color` colored list [=always/never/auto]

`ls -d` list directories - with ' */'

`ls -F` add one char of */=>@| to enteries

`ls -i` list file's inode index number

`ls -l` list with long format - show permissions

`ls -la` list long format including hidden files

`ls -lh` list long format with readable file size

`ls -ls` list with long format with file size

`ls -r` list in reverse order

`ls -R` list recursively directory tree

`ls -s` list file size

`ls -S` sort by file size

`ls -t` sort by time & date

`ls -X` sort by extension name

Section 7.2: ls command with most used options

ls shows files and directories in present working directory. (if no arguments are passed.) (It doesn't show hidden files which starts with . by default.)

```
user@ubuntu14:/usr$ ls
bin  games  include  lib  lib32  local  sbin  share  src
```

To see all files (hidden files/folders also). Use `ls -a` OR `ls -all`

```
user@ubuntu14:/usr$ ls -a
.  ..  bin  games  include  lib  lib32  local  sbin  share  src
```

To differentiate between files and folders and symbolic links and other, use `ls -F` OR `ls --classify`

```
user@ubuntu14:~$ ls -F
bash_profile_course  chat_apps/      Desktop/    Downloads/    foxitsoftware/
Public/     test/    bin/    ClionProjects/  Documents/    IDE/    Music/
Pictures/  Templates/  Videos/
```

Here, ending characters are used to distinguish files and folders.

“/” suggest directory.

“*”suggest executables.

“@” suggest symbolic links.

To get more details about the files and directories, use `ls -l`

```
user@ubuntu14:~/example$ ls -l
total 6464

-rw-r--r-- 1 dave dave      41 Dec 24 12:19 Z.txt
drwxr-xr-x 2 user group    4096 Dec 24 12:00 a_directory
-rw-r--r-- 1 user group       6 Dec 24 12:01 a_file
lrwxrwxrwx 1 user group       6 Dec 24 12:04 a_link -> a_file
-rw-r--r-- 1 user group       6 Dec 24 12:03 a_newer_file
-rw-r----- 1 user group 6586816 Dec 24 12:07 big.zip
```

In this example, the total size of the contents is 6460KB.

Then there is an entry for each file/directory in alphabetical order with upper case before lower case.

The first character is the type (e.g. d - directory, l - link).

The next 9 characters show the permissions for the user, group and other.

This is followed by the number of hard links, then the owner's name and group.

The next field is the size in bytes. This can be displayed in a human friendly form by adding the `-h` option e.g. 6586816 is displayed as 6.3M

There then follows a timestamp (usually the modification time).

The final field is the name. Note: links also show the target of the link.

Chapter 8: File Compression with 'tar' command

Common Options	-
-c --create	Create a new archive.
-x --extract	Extract files from an archive.
-t --list	List the contents of an archive.
-f --file=*ARCHIVE*	Use archive file or dir *ARCHIVE*.
-v --verbose	Verbosely list files processed.
Compression Options	-
-a --auto-compress	Use archive suffix to determine the compression program.
-j --bzip2	Filter the archive through bzip2.
-J --xz --lzma	Filter the archive through xz.
-z --gzip	Filter the archive through gzip.

Section 8.1: Compress a folder

This creates a simple archive of a folder :

```
tar -cf ./my-archive.tar ./my-folder/
```

Verbose output shows which files and directories are added to the archive, use the -v option:

```
tar -cvf ./my-archive.tar ./my-folder/
```

For archiving a folder compressed 'gzip', you have to use the -z option :

```
tar -czf ./my-archive.tar.gz ./my-folder/
```

You can instead compress the archive with 'bzip2', by using the -j option:

```
tar -cjf  ./my-archive.tar.bz2 ./my-folder/
```

Or compress with 'xz', by using the -J option:

```
tar -cJf  ./my-archive.tar.xz ./my-folder/
```

Section 8.2: Extract a folder from an archive

There is an example for extract a folder from an archive in the current location :

```
tar -xf archive-name.tar
```

If you want to extract a folder from an archive to a specfic destination :

```
tar -xf archive-name.tar -C ./directory/destination
```

Section 8.3: List contents of an archive

List the contents of an archive file without extracting it:

```
tar -tf archive.tar.gz
Folder-In-Archive/
Folder-In-Archive/file1
Folder-In-Archive/Another-Folder/
Folder-In-Archive/Another-Folder/file2
```

Section 8.4: List archive content

There is an example of listing content :

```
tar -tvf archive.tar
```

The option `-t` is used for the listing. For listing the content of a tar.gz archive, you have to use the `-z` option anymore :

```
tar -tzvf archive.tar.gz
```

Section 8.5: Compress and exclude one or multiple folder

If you want to extract a folder, but you want to exclude one or several folders during the extraction, you can use the `--exclude` option.

```
tar -cf archive.tar ./my-folder/ --exclude="my-folder/sub1" --exclude="my-folder/sub3"
```

With this folder tree :

```
my-folder/
   sub1/
   sub2/
   sub3/
```

The result will be :

```
./archive.tar
   my-folder/
      sub2/
```

Section 8.6: Strip leading components

To strip any number of leading components, use the --strip-components option:

```
 --strip-components=NUMBER
   strip NUMBER leading components from file names on extraction
```

For example to strip the leading folder, use:

```
tar -xf --strip-components=1 archive-name.tar
```

Chapter 9: Services

Section 9.1: List running service on Ubuntu

To get a list of the service on your system, you may run:

```
service --status-all
```

The output of `service --status-all` lists the state of services controlled by `System V`.

The `+` indicates the service is running, `-` indicates a stopped service. You can see this by running service SERVICENAME status for a + and - service.

Some services are managed by **Upstart**. You can check the status of all Upstart services with sudo initctl list. Any service managed by Upstart will also show in the list provided by service --status-all but will be marked with a ?.

ref: https://askubuntu.com/questions/407075/how-to-read-service-status-all-results

Section 9.2: Systemd service management

Listing services

- `systemctl` To list running services
- `systemctl --failed` To list failed services

Managing Targets (Similar to Runlevels in SysV)

- `systemctl get-default` To find the default target for your system
- `systemctl set-default <target-name>` To set the default target for your system

Managing services at runtime

- `systemctl start [service-name]` To start a service
- `systemctl stop [service-name]` To stop a service
- `systemctl restart [service-name]` To restart a service
- `systemctl reload [service-name]` To request service to reload its configuration
- `systemctl status [service-name]` To show current status of a service

Managing autostart of services

- `systemctl is-enabled [service-name]` To show whether a service is enabled on system boot
- `systemctl is-active [service-name]` To show whether a service is currently active(running)
- `systemctl enable [service-name]` To enable a service on system boot
- `systemctl disable [service-name]` To disable a service on system boot

Masking services

- `systemctl mask [service-name]` To mask a service (Makes it hard to start a service by mistake)
- `systemctl unmask [service-name]` To unmask a service

Restarting systemd

```
systemctl daemon-reload
```

Chapter 10: Managing Services

Section 10.1: Diagnosing a problem with a service

On systems using systemd, such as Fedora => 15, Ubuntu (Server and Desktop) >= 15.04, and RHEL/CentOS >= 7:

```
systemctl status [servicename]
```

...where `[servicename]` is the service in question; for example, `systemctl status sshd`.

This will show basic status information and any recent errors logged.

You can see further errors with `journalctl`. For example,`journalctl -xe` will load the last 1000 logged into a pager (like `less`), jumping to the end. You can also use `journalctl -f`, which will follow log messages as they come in.

To see logs for a particular service, use the `-t` flag, like this:

```
journalctl -f -t sshd
```

Other handy options include `-p` for priority (`-p warnings` to see only warnings and above), `-b` for "since last boot", and `-S` for "since" — putting that together, we might do

```
journalctl -p err -S yesterday
```

to see all items logged as errors since yesterday.

If journalctl is not available, or if you are following application error logs which do not use the system journal, the `tail` command can be used to show the last few lines of a file. A useful flag for tail is `-f` (for "follow"), which causes tail continue showing data as it gets appended to the file. To see messages from most services on the system:

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

Or, if the service is privileged, and may log sensitive data:

```
tail -f /var/log/secure
```

Some services have their own log files, a good example is `auditd`, the linux auditing daemon, which has its logs stored in `/var/log/audit/`. If you do not see output from your service in `/var/log/messages` try looking for service specific logs in `/var/log/`

Section 10.2: Starting and Stopping Services

On systems that use the System-V style init scripts, such as RHEL/CentOS 6:

```
service <service> start
```

```
service <service> stop
```

On systems using systemd, such as Ubuntu (Server and Desktop) >= 15.04, and RHEL/CentOS >= 7:

```
systemctl <service> dnsmasq
```

```
systemctl <service> dnsmasq
```

Section 10.3: Getting the status of a service

On systems that use the System-V style init scripts, such as RHEL/CentOS 6:

```
service <service> status
```

On systems using systemd, such as Ubuntu (Server and Desktop) >= 15.04, and RHEL/CentOS >= 7.0:

```
systemctl status <service>
```

Chapter 11: Modifying Users

Parameter	Details
username	The name of the user. Do not use capital letters, do not use dots, do not end it in dash, it must not include colons, no special characters. Cannot start with a number.

Section 11.1: Setting your own password

```
passwd
```

Section 11.2: Setting another user's password

Run the following as root:

```
passwd username
```

Section 11.3: Adding a user

Run the following as root:

```
useradd username
```

Section 11.4: Removing a user

Run the following as root:

```
userdel username
```

Section 11.5: Removing a user and its home folder

Run the following as root:

```
userdel -r username
```

Section 11.6: Listing groups the current user is in

```
groups
```

More detailed information about user and group numerical IDs can be found with the `id` command.

Section 11.7: Listing groups a user is in

```
groups username
```

More detailed information about user and group numerical IDs can be found with `id username`.

Chapter 12: LAMP Stack

LAMP (**L**inux **A**pache **M**ySQL **P**HP) consists of the Linux operating system as development environment, the Apache HTTP Server as web server, the MySQL relational database management system (RDBMS) as DB (**D**ata **B**ase) system, and the PHP programming language as Server side (Back End) programming language.

LAMP is used as a Open Source stack of technologies solution to web development area. Windows version of this stack is called WAMP (**W**indows **A**pache **M**ySQL **P**HP)

Section 12.1: Installing LAMP on Arch Linux

With this line we will install all the necessary packages in one step, and the last update:

```
pacman -Syu apache php php-apache mariadb
```

HTTP

Edit

`/etc/httpd/conf/httpd.conf`

Change `ServerAdmin you@example.com` as you need.

The folder of the WEB Pages by default is `ServerRoot "/etc/httpd"`. Directory must be set to the same folder, so change the line

```
<Directory "/etc/httpd">
```

This folder must have read and execution access, so

```
chmod o+x /etc/httpd
```

Change `AllowOverride from none (default) to All` so .htaccess will works.

Now you need the `~/public_html` folder for each user. (to get the root page of each user as http://localhost/~yourusername/. Unremark this line:

```
Include conf/extra/httpd-userdir.conf
```

Now as root you need to create the `~/public_html` for each user and change the access to (755) of each one.

```
chmod 755 /home
chmod 755 /home/username
chmod 755 /home/username/public_html
```

You can comment out this line if you want to use SSL:

```
LoadModule ssl_module modules/mod_ssl.so
```

If you need to use virtual domains, uncomment the line:

```
Include conf/extra/httpd-vhosts.conf
```

and in `/etc/httpd/conf/extra/httpd-vhosts.conf` you must to add all the virtual domains. (plus into `/etc/hosts` if you want to test those virtuals domains)

Edit `/etc/httpd/conf/extra/httpd-default.conf` and change **ServerSignature** to Off and **ServerToken** to Prod for hiding critical data

PHP

Edit: `/etc/httpd/conf/httpd.conf`

Comment out: `LoadModule mpm_event_module modules/mod_mpm_event.so`

Uncomment: `LoadModule mpm_prefork_module modules/mod_mpm_prefork.so`

As last item in the LoadModule list, add `LoadModule php7_module modules/libphp7.so`

As last item in the include list, add `Include conf/extra/php7_module.conf`

Edit `/etc/php/php.ini`

Uncomment `extension=mysqli.so` and `extension=pdo_mysql.so`

Change the timezone as you need, for example:

```
date.timezone = America/Argentina/Buenos_Aires, date.default_latitude = 0.0, date.default_longitude
= 0.0
```

MySQL

Run as root:

```
mysql_install_db --user=mysql --basedir=/usr --datadir=/var/lib/mysql
```

Now you have the root of the MySQL Server.

Start MySQL daemon:

```
systemctl enable mysqld
systemctl start mysqld
```

At last, run:

```
sh /usr/bin/mysql_secure_installation
```

That all to get a web server ready to be customized as you need.

Section 12.2: Installing LAMP on Ubuntu

Install apache:

```
sudo apt-get install apache2
```

Install MySql:

```
sudo apt-get install mysql-server
```

Install PHP:

```
sudo apt-get install php5 libapache2-mod-php5
```

Restart system:

```
sudo systemctl restart apache2
```

Check PHP installation:

```
php -r 'echo "\n\nYour PHP installation is working fine.\n\n\n";'
```

Section 12.3: Installing LAMP stack on CentoOS

Install Apache Web Server

First step is to install web server Apache.

```
sudo yum -y install httpd
```

Once it is installed, enable (to run on startup) and start Apache web server service.

```
sudo systemctl enable --now httpd
```

Point your browser to:

http://localhost

You will see the default Apache web server page.

Install MariaDB Server

Second step is to install MariaDB:

```
sudo yum -y install mariadb-server
```

Then start and enable (on startup) the MariaDB server:

```
sudo systemctl enable --now mariadb
```

As needed, use **mysql_secure_installation** to secure your database.

This script will allow you to do the following:

- Change the root user's password
- Remove test databases
- Disable remote access

Install PHP

```
sudo yum -y install php php-common
```

Then restart Apache's httpd service.

```
sudo systemctl restart httpd
```

To test PHP, create a file called **index.php** in **/var/www/html**.

Then add the following line to the file:

Then point your browser to:

http://localhost/index.php

You should see information related to your server. If you do not, ensure that php is for sure installed correctly by running the following command:

```
php --version
```

If you receive something like:

```
PHP 5.4.16 (cli) (built: Nov 6 2016 00:29:02) Copyright (c) 1997-2013 The PHP Group
```

Then PHP is installed correctly. If this is the case, please ensure that you've restarted your web server.

Chapter 13: tee command

Options	Description
-a, --append	Append to the given FILEs. Do not overwrite.
-i, --ignore-interrupts	Ignore interrupt signals.
--help	Display a help message, and exit.
--version	Display version information, and exit.

tee - read from standard input and write to standard output and files.

The tee command is named after the T-splitter in plumbing, which splits water into two directions and is shaped like an uppercase T.

tee copies data from standard input to each FILE, and also to standard output. In effect, tee duplicates its input, routing it to multiple outputs at once.

Section 13.1: Write output to stdout, and also to a file

The following command displays output only on the screen (stdout).

```
$ ls
```

The following command writes the output only to the file and not to the screen.

```
$ ls > file
```

The following command (with the help of `tee` command) writes the output both to the screen (stdout) and to the file.

```
$ ls | tee file
```

Section 13.2: Write output from the middle of a pipe chain to a file and pass it back to the pipe

You can also use `tee` command to store the output of a command in a file and redirect the same output to another command.

The following command will write current crontab entries to a file `crontab-backup.txt` and pass the crontab entries to `sed` command, which will do the substituion. After the substitution, it will be added as a new cron job.

```
$ crontab -l | tee crontab-backup.txt | sed 's/old/new/' | crontab –
```

Section 13.3: write the output to multiple files

You can pipe your output to multiple files (including your terminal) by using `tee` like this:

```
$ ls | tee file1 file2 file3
```

Section 13.4: Instruct tee command to append to the file

By default `tee` command overwrites the file. You can instruct `tee` to append to the file using the –a option as shown

below.

```
$ ls | tee –a file
```

Chapter 14: Secure Shell (SSH)

A secure shell is used to remotely access a server from a client over an encrypted connection. OpenSSH is used as an alternative to Telnet connections that achieve remote shell access but are unencrypted. The OpenSSH Client is installed on most GNU/Linux distributions by default and is used to connect to a server. These examples show use how to use the SSH suite to for accept SSH connections and connecting to another host.

Section 14.1: Connecting to a remote server

To connect to a server we must use SSH on the client as follows,

```
# ssh -p port user@server-address
```

- **port** - The listening ssh port of the server (default port 22).
- **user** - Must be an existing user on the server with SSH privileges.
- **server address** - The IP/Domain of the server.

For a real world example lets pretend that you're making a website. The company you chose to host your site tells you that the server is located at **web-servers.com** on a custom port of **2020** and your account name **usr1** has been chosen to create a user on the server with SSH privileges. In this case the SSH command used would be as such

```
# ssh -p 2020 usr1@web-servers.com
```

If account name on the remote system is the same as the one one the local client you may leave the user name off. So if you are `usr1` on both systems then you my simply use `web-servers.com` instead of `usr1@web-servers.com`.

When a server you want to connect to is not directly accessible to you, you can try using ProxyJump switch to connect to it through another server which is accessible to you and can connect to the desired server.

```
# ssh -J usr1@10.0.0.1:2020 usr2@10.0.0.2 -p 2222
```

This will let you connect to the server 10.0.0.2 (running ssh on port 2222) through server at 10.0.0.1 (running ssh on port 2020). You will need to have accounts on both servers of course. Also note that the -J switch is introduced in OpenSSH version 7.3.

Section 14.2: Installing OpenSSH suite

Both connecting to a remove SSH server and accepting SSH connections require installation of `openssh`

Debian:

```
# apt-get install openssh
```

Arch Linux:

```
# pacman -S openssh
```

Yum:

```
# yum install openssh
```

Section 14.3: Configuring an SSH server to accept connections

First we must edit the SSH daemon config file. Though under different Linux distributions this may be located in different directories, usually it is stored under `/etc/ssh/sshd_config`

Use your text editor to change the values set in this file, all lines starting with # are commented out and must have this character removed to take any effect. A list of recommendations follow as such.

```
Port (chose a number between 0 - 65535, normaly greater than four digits)
PasswordAuthentication yes
AllowUsers    user1 user2 ...etc
```

Note that it is preferable to disable password logins all together and use SSH Keys for improved security as explained in this document.

Section 14.4: Passwordless connection (using a key pair)

First of all you'll need to have a key pair. If you don't have one yet, take a look at the 'Generate public and private key topic'.

Your key pair is composed by a private key (id_rsa) and a public key (id_rsa.pub). All you need to do is to copy the public key to the remote host and add its contents to the `~/.ssh/authorized_keys` file.

One simple way to do that is:

```
ssh <user>@<ssh-server> 'cat >> ~/.ssh/authorized_keys' < id_rsa.pub
```

Once the public key is properly placed in your user's home directory, you just need to login using the respective private key:

```
ssh <user>@<ssh-server> -i id_rsa
```

Section 14.5: Generate public and private key

To generate keys for SSH client:

```
ssh-keygen [-t rsa | rsa1 | dsa ] [-C <comment>] [-b bits]
```

For example:

```
ssh-keygen -t rsa -b 4096 - C myemail@email.com
```

Default location is `~/.ssh/id_rsa` for private and `~/.ssh/id_rsa.pub` for public key.

For more info, please visit man.openbsd.org

Section 14.6: Disable ssh service

This will disable the SSH server side service, as if needed this will insure that clients cannot connect via ssh

Ubuntu

```
sudo service ssh stop
```

```
sudo systemctl disable sshd.service
```

Debian

```
sudo /etc/init.d/ssh stop
sudo systemctl disable sshd.service
```

Arch Linux

```
sudo killall sshd
sudo systemctl disable sshd.service
```

Chapter 15: SCP

Section 15.1: Secure Copy

scp command is used to securely copy a file to or from a remote destination. If the file is in current working directly only filename is sufficient else full path is required which included the remote hostname e.g. remote_user@some_server.org:/path/to/file

Copy local file in your CWD to new directory

```
scp localfile.txt /home/friend/share/
```

Copy remote file to you current working directory

```
scp rocky@arena51.net:/home/rocky/game/data.txt ./
```

Copy file from one remote location to another remote location

```
scp mars@universe.org:/beacon/light/bitmap.conf jupiter@universe.org:/beacon/night/
```

To copy directory and sub-directories use '-r' recursive option to scp

```
scp -r user@192.168.0.4:~/project/* ./workspace/
```

Section 15.2: Basic Usage

```
# Copy remote file to local dir
scp user@remotehost.com:/remote/path/to/foobar.md /local/dest

# Copy local file to remote dir
scp foobar.md user@remotehost.com:/remote/dest

# Key files can be used (just like ssh)
scp -i my_key.pem foobar.md user@remotehost.com:/remote/dest
```

Chapter 16: GnuPG (GPG)

GnuPG is a sophisticated key management system which allows for secure signing or encrypting data. GPG is a command-line tool used to create and manipulate GnuPG keys.

GnuPG is most widely used for having SSH (Secure Shell) connections without password or any means of interactive authentication, which improves security level significantly.

Following sections describe ways to create, use, and maintain security of GnuPG keys.

Section 16.1: Exporting your public key

In order for your public-private keypair to be of use, you must make your public key freely available to others. Be sure that you are working with your public key here since you should *never* share your private key. You can export your public key with the following command:

```
gpg —armor —export EMAIL_ADDRESS > public_key.asc
```

where EMAIL_ADDRESS is the email address associated with the key

Alternately, you can upload your public key to a public key server such as keys.gnupg.net so that others can use it. To do so, enter the following in a terminal:

```
gpg —list-keys
```

Then, search for the 8-digit string (the primary ID) associated with the key you want to export. Then, issue the command:

```
gpg —send-keys PRIMARY_ID
```

where PRIMARY_ID is the actual ID of that key.

Now, the public key has been uploaded to the key server and is publicly available.

Section 16.2: Create and use a GnuPG key quickly

Install haveged (example `sudo apt-get install haveged`) to speed up the random byte process. Then:

```
gpg --gen-key
gpg --list-keys
```

outputs:

```
pub   2048R/NNNNNNNN 2016-01-01
uid                  Name <name@example.com>
sub   2048R/xxxxxxxx 2016-01-01
```

Then publish:

```
gpg --keyserver pgp.mit.edu --send-keys NNNNNNNN
```

Then plan to revoke: https://www.hackdiary.com/2004/01/18/revoking-a-gpg-key/

Chapter 17: Network Configuration

This document covers TCP/IP networking, network administration and system configuration basics. Linux can support multiple network devices. The device names are numbered and begin at zero and count upwards. For example, a computer with two NICs will have two devices labeled eth0 and eth1.

Section 17.1: Local DNS resolution

File: `/etc/hosts` contains a list of hosts that are to be resolved locally(not by DNS)

Sample contents of the file:

```
127.0.0.1         your-node-name.your-domain.com  localhost.localdomain  localhost
XXX.XXX.XXX.XXX   node-name
```

The file format for the hosts file is specified by RFC 952

Section 17.2: Configure DNS servers for domain name resolution

File: `/etc/resolv.conf` contains a list of DNS servers for domain name resolution

Sample contents of the file:

```
nameserver 8.8.8.8 # IP address of the primary name server
nameserver 8.8.4.4 # IP address of the secondary name server
```

In case internal DNS server you can validate if this server resolve DNS names properly using `dig` command:

```
$ dig google.com @your.dns.server.com +short
```

Section 17.3: See and manipulate routes

Manipulate the IP routing table using `route`

Display routing table

```
$ route # Displays list or routes and also resolves host names
$ route -n # Displays list of routes without resolving host names for faster results
```

Add/Delete route

Option	Description
`add` or `del`	Add or delete a route
`-host x.x.x.x`	Add route to a single host identified by the IP address
`-net x.x.x.x`	Add route to a network identified by the network address
`gw x.x.x.x`	Specify the network gateway
`netmask x.x.x.x`	Specify the network netmask
`default`	Add a default route

Examples:

- add route to a host `$ route add -host x.x.x.x eth1`
- add route to a network `$ route add -net 2.2.2.0 netmask 255.255.255.0 eth0`
- Alternatively, you could also use cidr format to add a route to network `route add -net 2.2.2.0/24 eth0`
- add default gateway `$ route add default gw 2.2.2.1 eth0`
- delete a route `$ route del -net 2.2.2.0/24`

Manipulate the IP routing table using `ip`

Display routing table

```
$ ip route show # List routing table
```

Add/Delete route

Option	Description
`add` or `del` or `change` or `append` or `replace`	Change a route
`show` or `flush`	the command displays the contents of the routing tables or remove it
`restore`	restore routing table information from stdin
`get`	this command gets a single route to a destination and prints its contents exactly as the kernel sees it

Examples:

- Set default gateway to 1.2.3.254 `$ ip route add default via 1.2.3.254`
- Adds a default route (for all addresses) via the local gateway 192.168.1.1 that can be reached on device eth0 `$ ip route add default via 192.168.1.1 dev eth0`

Section 17.4: Configure a hostname for some other system on your network

You can configure your Linux (or macOS) system in order to tie in an identifier **`<hostname>`** to some other system's IP address in your network. You can configure it:

- Systemwide. You should modify the */etc/hosts* file. You just have to add to that file a new line containing:

    1. the remote system's IP address **`<ip_rem>`**,
    2. one or more blank spaces, and
    3. the identifier **`<hostname>`**.

- For a single user. You should modify the *~/.hosts* file --- you-d have to create it. It is not as simple as for systemwide. Here you can see an explanation.

For instance, you could add this line using the **`cat`** Unix tool. Suppose that you want to make a **`ping`** to a PC in yout local network whose IP address is 192.168.1.44 and you want to refer to that IP address just by `remote_pc`. Then you must write on your shell:

```
$ sudo cat 192.168.1.44 remote_pc
```

Then you can make that ping just by:

```
$ ping remote_pc
```

Section 17.5: Interface details

Ifconfig

List all the interfaces available on the machine

```
$ ifconfig -a
```

List the details of a specific interface

Syntax: `$ ifconfig <interface>`

Example:

```
$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr xx:xx:xx:xx:xx:xx
          inet addr:x.x.x.x  Bcast:x.x.x.x  Mask:x.x.x.x
          inet6 addr: xxxx::xxx:xxxx:xxxx:xxxx/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4426618 errors:0 dropped:1124 overruns:0 frame:0
          TX packets:189171 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:382611580 (382.6 MB)  TX bytes:36923665 (36.9 MB)
          Interrupt:16 Memory:fb5e0000-fb600000
```

Ethtool - query the network driver and hardware settings

Syntax: `$ ethtool <interface>`

Example:

```
$ ethtool eth0
```

```
Settings for eth0:
Supported ports: [ TP ]
Supported link modes:   10baseT/Half 10baseT/Full
100baseT/Half 100baseT/Full
1000baseT/Full
Supported pause frame use: No
Supports auto-negotiation: Yes
Advertised link modes:  10baseT/Half 10baseT/Full
100baseT/Half 100baseT/Full
1000baseT/Full
Advertised pause frame use: No
Advertised auto-negotiation: Yes
Speed: 1000Mb/s
Duplex: Full
Port: Twisted Pair
PHYAD: 1
Transceiver: internal
Auto-negotiation: on
MDI-X: on (auto)
Supports Wake-on: pumbg
Wake-on: g
Current message level: 0x00000007 (7)
drv probe link
Link detected: yes
```

ip - show / manipulate routing, devices, policy routing and tunnels

Syntax: `$ ip { link | ... | route | macsec }` (please see `man ip` for full list of objects)

Examples

List network interfaces

```
$ ip link show
```

Rename interface eth0 to wan

```
$ ip link set dev eth0 name wan
```

Bring interface eth0 up (or down)

```
$ ip link set dev eth0 up
```

List addresses for interfaces

```
$ ip addr show
```

Add (or del) ip and mask (255.255.255.0)

```
$ ip addr add 1.2.3.4/24 brd + dev eth0
```

Section 17.6: Adding IP to an interface

An IP address to an interface could be obtained via DHCP or Static assignment

DHCP If you are connected to a network with a DHCP server running, `dhclient` command can get an IP address for your interface

```
$ dhclient <interface>
```

or alternatively, you could make a change to the `/etc/network/interfaces` file for the interface to be brought up on boot and obtain DHCP IP

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

Static configuration(Permanent Change) using `/etc/network/interfaces` file

If you want to statically configure the interface settings(permanent change), you could do so in the `/etc/network/interfaces` file.

Example:

```
auto eth0 # Bring up the interface on boot
iface eth0 inet static
    address 10.10.70.10
    netmask 255.255.0.0
    gateway 10.10.1.1
    dns-nameservers 10.10.1.20
    dns-nameservers 10.10.1.30
```

These changes persist even after system reboot.

Static configuration(Temporary change) using `ifconfig` utility

A static IP address could be added to an interface using the **`ifconfig`** utility as follows

```
$ ifconfig <interface> <ip-address>/<mask> up
```

Example:

```
$ ifconfig eth0 10.10.50.100/16 up
```

Chapter 18: Midnight Commander

Midnight Commander or mc is a console file manager. This topic includes the descripton of it's functionalities and examples and tips of how to use it to it's full potential.

Section 18.1: Midnight Commander function keys in browsing mode

Here is a list of actions which can be triggered in the Midnight Commander filesystem browsing mode by using function keys on your keyboard.

`F1` Displays help

`F2` Opens user menu

`F3` Displays the contents of the selected file

`F4` Opens the selected file in the internal file editor

`F5` Copies the selected file to the directory open in the second panel

`F6` Moves the selected file to the directory open in the second panel

`F7` Makes a new directory in the directory open in the current panel

`F8` Deletes the selected file or directory

`F9` Focuses to the main menu on the top of the screen

`F10` Exits mc

Section 18.2: Midnight Commander function keys in file editing mode

Midnight Commander has a built in editor which is started by `F4` function key when over the desired file in the browse mode. It can also be invoked in standalone mode by executing

```
mcedit <filename>
```

Here is a list of actions which can be triggered in the edit mode.

`F1` Displays help

`F2` Saves current file

`F3` Marks the start of the text selection. Move cursor any direction to select. Second hit marks the end of the selection.

`F4` Brings up the text search/replace dialog

`F5` Copies selected text to the cursor location (copy/paste)

`F6` Moves selected text to the cursor location (cut/paste)

`F7` Brings up the text search dialog

F8 Deletes selected text

F9 Focuses to the main menu on the top of the screen

F10 Exits the editor

Chapter 19: Change root (chroot)

Change root (chroot) is an operation that changes the apparent root directory for the current running process and their children. A program that is run in such a modified environment cannot access files and commands outside that environmental directory tree.

Section 19.1: Requirements

- root privileges
- another working Linux environment,such as Live CD boot or an existing distribution
- matching environment architectures of `chroot` source and destination (check current environment architecture with `uname -m`)
- kernel modules which you may need in `chroot` environment must be loaded (for example, with `modprobe`)

Section 19.2: Manually changing root in a directory

1. Ensure you met all requirements, as per Requirements
2. Mount the temporary API filesystems:

```
cd /location/of/new/root
mount -t proc proc proc/
mount --rbind /sys sys/
mount --rbind /dev dev/
mount --rbind /run run/ (optionally)
```

3. If you need to use an internet connection in the chroot environment, copy over the DNS details:

```
cp /etc/resolv.conf etc/resolv.conf
```

4. Change root into /location/of/new/root, specifying the shell (`/bin/bash` in this example):

```
chroot /location/of/new/root /bin/bash
```

5. After chrooting it may be necessary to load the local bash configuration:

```
source /etc/profile
source ~/.bashrc
```

6. Optionally, create a unique prompt to be able to differentiate your chroot environment:

```
export PS1="(chroot) $PS1"
```

7. When finished with the chroot, you can exit it via:

```
exit
```

8. Unmount the temporary file systems:

```
cd /
umount --recursive /location/of/new/root
```

Section 19.3: Reasons to use chroot

Changing root is commonly done for performing system maintenance on systems where booting and/or logging in is no longer possible.

Common examples are:

- reinstalling the bootloader
- rebuilding the initramfs image
- upgrading or downgrading packages
- resetting a forgotten password
- building software in a clean root environment

Chapter 20: Package Managers

Section 20.1: How to update packages with the apt package manager

The **A**dvanced **P**ackage **T**ool, aptly named the 'apt' package manager can handle the installation and removal of software on the Debian, Slackware, and other Linux Distributions. Below are some simple examples of use:

update
This option retrieves and scans the Packages.gz files, so that information about new and updated packages is available. To do so, enter the following command:

```
sudo apt-get update
```

upgrade
This option is used to install the newest versions of all packages currently installed on the system. Packages currently installed with new versions available are retrieved and upgraded; under no circumstances are currently installed packages removed, or packages not already installed retrieved and installed. To upgrade, enter the following command:

```
sudo apt-get upgrade
```

dist-upgrade
In addition to performing the function of upgrade, dist-upgrade also intelligently handles changing dependencies with new versions of packages. It will attempt to upgrade the most important packages at the expense of less important ones if necessary. To do so, enter the following command:

```
sudo apt-get dist-upgrade
```

Section 20.2: How to install a package with the pacman package manager

In order to search for packages in the databse, searching both in packages' names and descriptions:

```
pacman -Ss string1 string2 ...
```

To install a single package or list of packages (including dependencies), issue the following command:

```
sudo pacman -S package_name1 package_name2 ...
```

source

Section 20.3: How to update packages with the pacman package manager

To update a specific program:

```
sudo pacman -S <programName>
```

To update entire the system:

```
sudo pacman -Syu
```

Section 20.4: How to update packages with yum

Yellowdog **U**pdater, **M**odified, one of the last remaining vestiges of Yellow Dog Linux, is the package manager used by Red Hat, Fedora, and CentOS systems and their derivatives. It can handle the installation and removal of software packaged as **rpms** for these Linux distributions. Below are some simple examples of use:

search
This command will attempt to locate software packages in the configured software repositories that match the given search criteria, and display the name / version / repository location of the matches it finds. To use it, enter the following command:

```
yum search <queryString>
```

install
This command will attempt to locate and install the named software from the configured software repositories, recursively locating and installing any needed prerequisite software as well. To use it, enter the following command:

```
sudo yum install <packageName>
```

update
This option is used to install the newest versions of all packages currently installed on the system. Packages currently installed with new versions available are retrieved and upgraded; new prerequisites are also retrieved and installed as necessary, and replaced or obsoleted packages are removed. To upgrade, enter the following command:

```
sudo yum update
```

Unlike **apt**, most **yum** commands will also automatically check for updates to repository metadata if a check has not been done recently (or if forced to do so) and will retrieve and scan updated metadata so that information about new and updated packages is available before the requested operation is performed.

Chapter 21: Compiling the Linux kernel

Section 21.1: Compilation of Linux Kernel on Ubuntu

> **Warning:** be sure you have at least 15 GB of free disk space.

Compilation in Ubuntu >=13.04

Option A) Use Git

Use git if you want to stay in sync with the latest Ubuntu kernel source. Detailed instructions can be found in the Kernel Git Guide. The git repository does not include necessary control files, so you must build them by:

```
fakeroot debian/rules clean
```

Option B) Download the source archive

Download the source archive - This is for users who want to rebuild the standard Ubuntu packages with additional patches. Use a follow command to install the build dependencies and extract the source (to the current directory):

1. Install the following packages:

```
sudo apt-get build-dep linux-image-`uname -r`
```

Option C) Download the source package and build

This is for users who want to modify, or play around with, the Ubuntu-patched kernel source.

1. Retrieve the latest kernel source from kernel.org.
2. Extract the archive to a directory and `cd` into it:

```
tar xf linux-*.tar.xz
cd linux-*
```

3. Build the ncurses configuration interface:

```
make menuconfig
```

4. To accept the default configuration, press → to highlight `< Exit >` and then Return.
5. Press Return again to save the configuration.
6. Use `make` to build the kernel:

```
make
```

   Note that you can use the `-jem>` flag to compile files in parallel and take advantage of multiple cores.

The compressed kernel image can be found at `arch/[arch]/boot/bzImage`, where `[arch]` is equal to `uname -a`.