

ESSENTIAL LINUX TOOLKIT

37 COMMANDS
YOU SHOULD
KNOW

LINUX

Dave McKay

Linux Essential Toolkit

37 Commands You Should Know

Dave McKay

©2019 by LifeSavvy Media. All rights reserved. No part of this book may be reproduced in any form or by any electronic or mechanical means without permission in writing from the publisher, except by a reviewer, who may quote brief passages in a review.

Cover Photo by [profit image - stock.adobe.com](https://www.adobe.com/stock/)



Contents

The Essential Toolkit for the Terminal	1
1. alias	2
2. cat.....	3
3. cd.....	4
4. chmod	5
5. chown.....	6
6. curl	7
7. df	8
8. diff	9
9. echo.....	10
10. exit.....	11
11. find	11
12. finger	12
13. free	12
14. grep	12
15. groups	14
16. gzip	14
17. head	15
18. history	16
19. kill	16
20. less	17
21. ls	18
22. man	19
23. mkdir	20
24. mv.....	21
25. passwd	22
26. ping.....	22
27. ps.....	23
28. pwd	24
29. shutdown	24
30. SSH	26
31. sudo.....	27

32. tail27

33. tar28

34. top30

35. uname32

36. w33

37. whoami33

That's Your Toolkit33

The Essential Toolkit for the Terminal

Are you new to Linux or just a little rusty? Here are all the commands you'll need to know. Think of this as an essential reference for the Linux terminal. This applies to the macOS command line, too.

Linux includes a large number of commands, but we've chosen 37 of the most important ones to present here. Learn these commands, and you'll be much more at home at the Linux command prompt.

The below list is presented in alphabetical order. A command's position in the list is not representative of its usefulness or simplicity. For the final word on a command's usage, refer to its man pages. The `man` command is in our list, of course---it's short for "manual."

1. alias

The `alias` command lets you give your own name to a command or sequence of commands. You can then type your short name, and the shell will execute the command or sequence of commands for you.

```
alias cls=clear
```

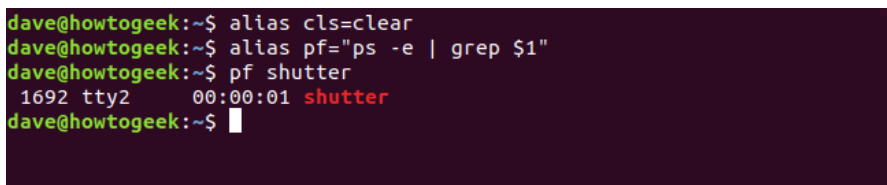
This sets up an alias called `cls`. It will be another name for `clear`. When you type `cls`, it will clear the screen just as though you had typed `clear`. Your alias saves a few keystrokes, sure. But, if you frequently move between Windows and Linux command line, you can find yourself typing the Windows `cls` command on a Linux machine that doesn't know what you mean. Now it will know.

Aliases can be much more intricate than that simple example. Here's an alias called `pf` (for process find) that is just a little more complex. Note the use of quotation marks around the command sequence. This is required if the command sequence has spaces in it. This alias uses the `ps` command to list the running processes and then [pipes](#) them through the `grep` command. The `grep` command looks for entries in the output from `ps` that match the command line parameter `$1`.

```
alias pf="ps -e | grep $1"
```

If you wanted to discover the process ID (PID) of the `shutter` process--or to find out if `shutter` was even running---you could use the alias like this. Type `pf`, a space, and the name of the process you are interested in:

```
pf shutter
```



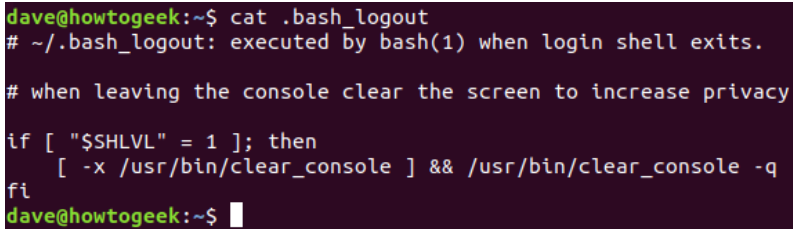
```
dave@howtogeek:~$ alias cls=clear
dave@howtogeek:~$ alias pf="ps -e | grep $1"
dave@howtogeek:~$ pf shutter
1692 tty2    00:00:01  shutter
dave@howtogeek:~$
```

Aliases defined on the command line will die with the terminal window. When you close it, they are gone. To make your aliases always be available to you, add them to the `.bash_aliases` file in your home directory.

2. cat

The `cat` command (short for "concatenate") lists the contents of files to the terminal window. This is faster than opening the file in an editor, and there's no chance you can accidentally alter the file. To read the contents of your `.bash_log_out` file, type the following command while the home directory is your current working directory, as it is by default:

```
cat .bash_logout
```



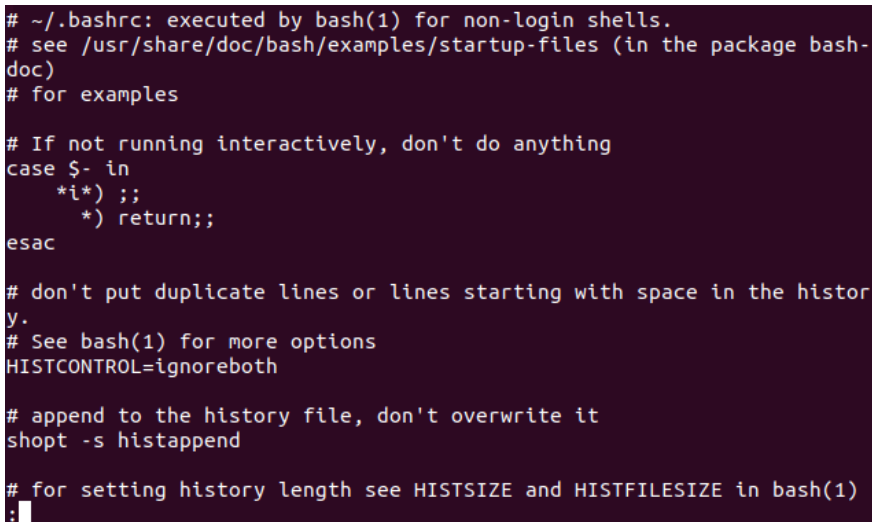
```
dave@howtogeek:~$ cat .bash_logout
# ~/.bash_logout: executed by bash(1) when login shell exits.

# when leaving the console clear the screen to increase privacy

if [ "$SHLVL" = 1 ]; then
    [ -x /usr/bin/clear_console ] && /usr/bin/clear_console -q
fi
dave@howtogeek:~$
```

With files longer than the number of lines in your terminal window, the text will whip past too fast for you to read. You can pipe the output from `cat` through `less` to make the process more manageable. With `less` you can scroll forward and backward through the file using the Up and Down Arrow keys, the PgUp and PgDn keys, and the Home and End keys. Type `q` to quit from `less`.

```
cat .bashrc | less
```



```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
    *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
:
```

3. cd

The `cd` command changes your current directory. In other words, it moves you to a new place in the filesystem.

If you are changing to a directory that is within your current directory, you can simply type `cd` and the name of the other directory.

```
cd work
```

If you are changing to a directory elsewhere within the filesystem directory tree, provide the path to the directory with a leading `/`.

```
cd /usr/local/bin
```

To quickly return to your home directory, use the `~` (tilde) character as the directory name.

```
cd ~
```

```
dave@howtogeek:~$ cd work
dave@howtogeek:~/work$ cd /usr/local/bin
dave@howtogeek:/usr/local/bin$ cd ~
dave@howtogeek:~$
```

Here's another trick: You can use the double dot symbol `..` to represent the parent of the current directory. You can type the following command to go up a directory:

```
cd ..
```

Imagine you are in a directory. The parent directory has other directories in it, as well as the directory you're currently in. To change into one of those other directories, you can use the `..` symbol to shorten what you have to type.

```
cd ../games
```

```
dave@howtogeek:/usr/local/bin$ cd ../games/
dave@howtogeek:/usr/local/games$
```


4. chmod

The `chmod` command [sets the file permissions flags](#) on a file or folder. The flags define who can read, write to or execute the file. When you list files with the `-l` (long format) option you'll see a string of characters that look like

```
-rwxrwxrwx
```

If the first character is a `-` the item is a file, if it is a `d` the item is a directory. The rest of the string is three sets of three characters. From the left, the first three represent the file permissions of the *owner*, the middle three represent the file permissions of the *group* and the rightmost three characters represent the permissions for *others*. In each set, an `r` stands for read, a `w` stands for write, and an `x` stands for execute.

If the `r`, `w`, or `x` character is present that file permission is granted. If the letter is not present and a `-` appears instead, that file permission is not granted.

One way to use `chmod` is to provide the permissions you wish to give to the owner, group, and others as a three-digit number. The leftmost digit represents the owner. The middle digit represents the group. The rightmost digit represents the others. The digits you can use and what they represent are listed here:

- **0:** No permission
- **1:** Execute permission
- **2:** Write permission
- **3:** Write and execute permissions
- **4:** Read permission
- **5:** Read and execute permissions
- **6:** Read and write permissions
- **7:** Read, write and execute permissions

Looking at our `example.txt` file, we can see that all three sets of characters are `rwx`. That means everyone has read, write and execute rights with the file.

To set the permission to be read, write, and execute (7 from our list) for the *owner*; read and write (6 from our list) for the *group*; and read and execute (5 from our list) for the *others* we'd need to use the digits 765 with the `chmod` command:

```
chmod -R 765 example.txt
```

```
dave@howtogeek:~/work$ ls -l example.txt
-rwxrwxrwx 1 dave dave 5957 Apr 23 15:58 example.txt
dave@howtogeek:~/work$
dave@howtogeek:~/work$ chmod 765 example.txt
dave@howtogeek:~/work$ ls -l example.txt
-rwxrw-r-x 1 dave dave 5957 Apr 23 15:58 example.txt
dave@howtogeek:~/work$
dave@howtogeek:~/work$ chmod 766 example.txt
dave@howtogeek:~/work$ ls -l example.txt
-rwxrw-rw- 1 dave dave 5957 Apr 23 15:58 example.txt
dave@howtogeek:~/work$
dave@howtogeek:~/work$
dave@howtogeek:~/work$
```

To set the permission to be read, write and execute (7 from our list) for the *owner*, and read and write (6 from our list) for the *group* and for the *others* we'd need to use the digits 766 with the `chmod` command:

```
chmod 766 example.txt
```

5. chown

The `chown` command allows you to change the owner and group owner of a file. Listing our `example.txt` file with `ls -l` we can see `dave dave` in the file description. The first of these indicates the name of the file owner, which in this case is the user `dave`. The second entry shows that the name of the group owner is also `dave`. Each user has a default group created when the user is created. That user is the only member of that group. This shows that the file is not shared with any other groups of users.

You can use `chown` to change the owner or group, or both of a file. You must provide the name of the owner and the group, separated by a colon (:). You will need to use `sudo`. To retain `dave` as the owner of the file but to set `mary` as the group owner, use this command:

```
sudo chown dave:mary example.txt
```

```
dave@howtogeek:~/work$ ls -l
total 8
-rwxrw-rw- 1 dave dave 5957 Apr 23 15:58 example.txt
dave@howtogeek:~/work$
dave@howtogeek:~/work$ sudo chown dave:mary example.txt
dave@howtogeek:~/work$ ls -l
total 8
-rwxrw-rw- 1 dave mary 5957 Apr 23 15:58 example.txt
dave@howtogeek:~/work$
dave@howtogeek:~/work$ sudo chown mary:mary example.txt
dave@howtogeek:~/work$ ls -l
total 8
-rwxrw-rw- 1 mary mary 5957 Apr 23 15:58 example.txt
dave@howtogeek:~/work$
dave@howtogeek:~/work$ sudo chown dave:dave example.txt
dave@howtogeek:~/work$ ls -l
total 8
-rwxrw-rw- 1 dave dave 5957 Apr 23 15:58 example.txt
dave@howtogeek:~/work$
```

To change both the owner and the group owner to mary, you would use the following command:

```
sudo chown mary:mary example.txt
```

To change the file so that dave is once more the file owner and the group owner, use this command:

```
sudo chown dave:dave example.txt
```

6. curl

The `curl` command is a tool to retrieve information and files from Uniform Resource Locators (URLs) or internet addresses.

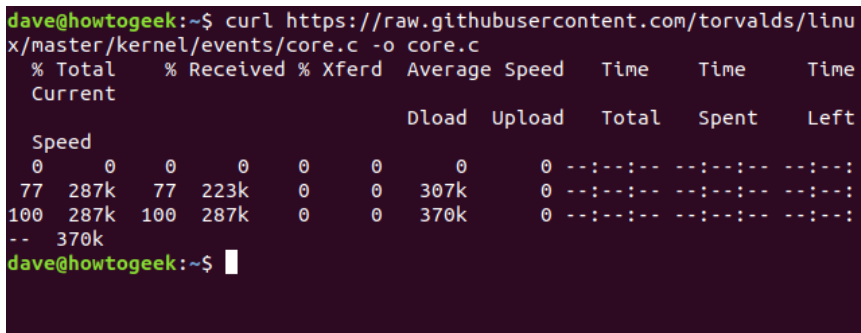
The `curl` command may not be provided as a standard part of your Linux distribution. Use `apt-get` to install this package onto your system if you're using Ubuntu or another Debian-based distribution. On other Linux distributions, use your Linux distribution's package management tool instead.

```
sudo apt-get install curl
```

Suppose you want to retrieve a single file from a GitHub repository. There is no officially supported way to this. You're forced to clone the entire repository. With `curl` however, we can retrieve the file we want on its own.

This command retrieves the file for us. Note that you need to specify the name of the file to save it in, using the `-o` (output) option. If you do not do this, the contents of the file are scrolled rapidly in the terminal window but not saved to your computer.

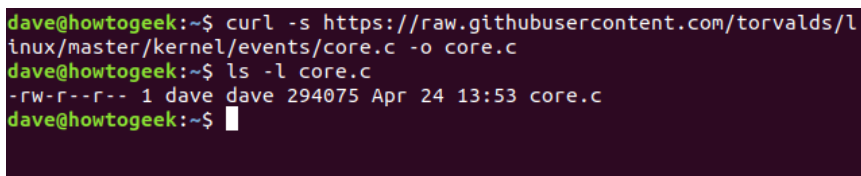
```
curl
https://raw.githubusercontent.com/torvalds/linux/master/kernel/events/core.c -o core.c
```



```
dave@howtogeek:~$ curl https://raw.githubusercontent.com/torvalds/linux/master/kernel/events/core.c -o core.c
% Total    % Received % Xferd  Average Speed   Time    Time     Time
  Current                                 Dload  Upload   Total   Spent    Left
  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
 77 287k   77 223k    0     0 307k      0 --:--:-- --:--:-- --:--:--
100 287k  100 287k    0     0 370k      0 --:--:-- --:--:-- --:--:--
-- 370k
dave@howtogeek:~$
```

If you don't want to see the download progress information, use the `-s` (silent) option.

```
curl -s
https://raw.githubusercontent.com/torvalds/linux/master/kernel/events/core.c -o core.c
```



```
dave@howtogeek:~$ curl -s https://raw.githubusercontent.com/torvalds/linux/master/kernel/events/core.c -o core.c
dave@howtogeek:~$ ls -l core.c
-rw-r--r-- 1 dave dave 294075 Apr 24 13:53 core.c
dave@howtogeek:~$
```

7. df

The `df` command [shows the size, used space, and available space](#) on the mounted filesystems of your computer.

Two of the most useful options are the `-h` (human readable) and `-x` (exclude) options. The human-readable option displays the sizes in Mb or Gb instead of in bytes. The exclude option allows you to tell `df` to discount filesystems you are not interested in. For example, the `squashfs` pseudo-filesystems that are created when you install an application with the `snap` command.

```
df -h -x squashfs
```

```
dave@howtogeek:~$ df -h -x squashfs
Filesystem      Size  Used Avail Use% Mounted on
udev            976M   0    976M   0% /dev
tmpfs           200M   1.6M  198M   1% /run
/dev/sda1       9.8G   6.7G  2.6G  73% /
tmpfs           997M   0    997M   0% /dev/shm
tmpfs           5.0M   4.0K   5.0M   1% /run/lock
tmpfs           997M   0    997M   0% /sys/fs/cgroup
VMShared        458G  339G  119G   74% /media/sf_VMShared
tmpfs           200M   28K   200M   1% /run/user/121
tmpfs           200M   48K   200M   1% /run/user/1000
dave@howtogeek:~$
```

8. diff

The `diff` command [compares two text files](#) and shows the differences between them. There are many options to tailor the display to your requirements.

The `-y` (side by side) option shows the line differences side by side. The `-w` (width) option lets you specify the maximum line width to use to avoid wraparound lines. The two files are called `alpha1.txt` and `alpha2.txt` in this example. The `--suppress-common-lines` prevents `diff` from listing the matching lines, letting you focus on the lines which have differences.

```
diff -y -W 70 alpha1.txt alpha2.txt --suppress-common-lines
```

```
dave@howtogeek:~$ diff -y -W 70 alpha1.txt alpha2.txt --suppress-common-lines
Delta
Lima
Uniform
>
>
>
dave@howtogeek:~$
```

9. echo

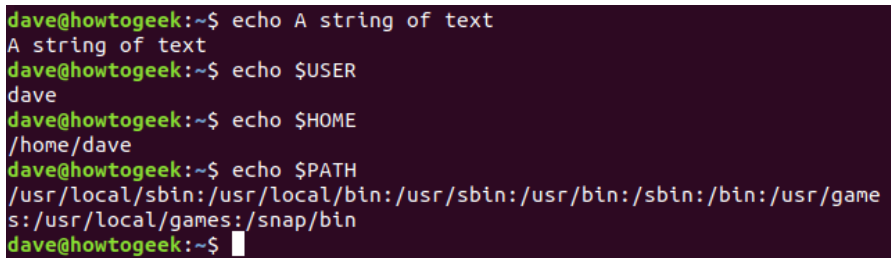
The `echo` command prints (echoes) a string of text to the terminal window.

The command below will print the words "A string of text" on the terminal window.

```
echo A string of text
```

The `echo` command can show the value of environment variables, for example, the `$USER`, `$HOME`, and `$PATH` environment variables. These hold the values of the name of the user, the user's home directory, and the path searched for matching commands when the user types something on the command line.

```
echo $USER
echo $HOME
echo $PATH
```

A terminal window with a dark purple background and green text. The prompt is 'dave@howtogeek:~\$'. The first command is 'echo A string of text' and the output is 'A string of text'. The second command is 'echo \$USER' and the output is 'dave'. The third command is 'echo \$HOME' and the output is '/home/dave'. The fourth command is 'echo \$PATH' and the output is '/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin'. The prompt is now 'dave@howtogeek:~\$' with a cursor.

```
dave@howtogeek:~$ echo A string of text
A string of text
dave@howtogeek:~$ echo $USER
dave
dave@howtogeek:~$ echo $HOME
/home/dave
dave@howtogeek:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
dave@howtogeek:~$
```

The following command will cause a bleep to be issued. The `-e` (escape code) option interprets the escaped a character as [a 'bell' character](#).

```
echo -e "\a"
```

The `echo` command is also invaluable in shell scripts. A script can use this command to generate visible output to indicate the progress or results of the script as it is executed.

10. exit

The `exit` command will close a terminal window, end the execution of a shell script, or log you out of an SSH remote access session.

```
exit
```

```
dave@howtogeek:~$ exit
```

11. find

Use the `find` command to track down files that you know exist if you can't remember where you put them. You must tell `find` where to start searching from and what it is looking for. In this example, the dot (`.`) matches the current folder and the `-name` option tells `find` to look for files with a name that matches the search pattern.

You can use wildcards, where `*` represents any sequence of characters and `?` represents any single character. We're using `*ones*` to match any file name containing the sequence "ones." This would match words like bones, stones, and lonesome.

```
find . -name *ones*
```

```
dave@howtogeek:~/Documents$ find . -name *ones*
./Ukulele/Random Songs/Im so lonesome I could cry.txt
./Ukulele/Random Songs/Get_Off_My_Cloud_Rolling_Stones.pdf
./Ukulele/Random Songs/Trail_of_the_Lonesome_Pine.pdf
./Ukulele/Ramones
./Ukulele/Possibles/Paint It Black (The Rolling Stones).pdf
dave@howtogeek:~/Documents$
dave@howtogeek:~/Documents$ find . -type f -name *ones*
./Ukulele/Random Songs/Im so lonesome I could cry.txt
./Ukulele/Random Songs/Get_Off_My_Cloud_Rolling_Stones.pdf
./Ukulele/Random Songs/Trail_of_the_Lonesome_Pine.pdf
./Ukulele/Possibles/Paint It Black (The Rolling Stones).pdf
dave@howtogeek:~/Documents$
dave@howtogeek:~/Documents$ find . -iname *wild*
./Ukulele/014 - Real Wild Child.odt
./Ukulele/Random Songs/Born_To_Be_Wild_Steppenwolf.pdf
./Ukulele/Random Songs/Wildwood Flower.pdf
./Ukulele/Random Songs/Real Wild Child.txt
./Ukulele/Random Songs/Wild Thing.pdf
dave@howtogeek:~/Documents$
```

As we can see, `find` has returned a list of matches. One of them is a directory called `Ramones`. We can tell `find` to restrict the search to files only. We do this using the `-type` option with the `f` parameter. The `f` parameter stands for files.

```
find . -type f -name *ones*
```

If you want the search to be case insensitive use the `-iname` (insensitive name) option.

```
find . -iname *wild*
```

12. finger

The `finger` command gives you a short dump of information about a user, including the time of the user's last login, the user's home directory, and the user account's full name.

```
dave@howtogeek:~$ finger mary
Login: mary                               Name: Mary Quinn
Directory: /home/mary                     Shell: /bin/bash
Last login Wed Apr 24 16:00 (EDT) on pts/2 from 192.168.4.27
No mail.
No Plan.
dave@howtogeek:~$
```

13. free

The `free` command gives you a summary of the memory usage with your computer. It does this for both the main Random Access Memory (RAM) and swap memory. The `-h` (human) option is used to provide human-friendly numbers and units. Without this option, the figures are presented in bytes.

```
free -h
```

```
dave@howtogeek:~/Documents$ free -h
              total        used        free      shared  buff/cache
available
Mem:           1.9G          1.1G          74M         321M         771M
             369M
Swap:           472M          6.5M          465M
```

14. grep

The `grep` utility searches for lines which contain a search pattern. When we looked at the `alias` command, we used `grep` to search through the output of another program, `ps`. The `grep` command can also search the contents of files. Here we're searching for the word "train" in all text files in the current directory.


```
grep train *.txt
```

The output lists the name of the file and shows the lines that match. The matching text is highlighted.

```
dave@howtogeek:~/work$ grep train *.txt
Blue Train.txt:It's callin' callin' callin', 'Come and get aboard the
train'
Blue Train.txt:Gonna ride a blue train, gonna ride a blue train
Blue Train.txt:Gonna ride a blue train, gonna ride a blue train
Im so lonesome I could cry.txt:The midnight train is whining low
Orange Blossom Special.txt:he's the fastest train on the line
Orange Blossom Special.txt:She's the fastest train on the line
Train Kept a Rolling.txt:I caught a train, I met a dame.
Train Kept a Rolling.txt:Well, the train kept a-rollin' all night long
.
Train Kept a Rolling.txt:The train kept a-rollin' all night long.
Train Kept a Rolling.txt:The train kept a-movin' all night long.
Train Kept a Rolling.txt:The train kept a-rollin' all night long.
Train Kept a Rolling.txt:We got off the train at El Paso,
Train Kept a Rolling.txt:Well, the train kept a-rollin' all night long
.
Train Kept a Rolling.txt:The train kept a-rollin' all night long.
Train Kept a Rolling.txt:The train kept a-movin' all night long.
Train Kept a Rolling.txt:The train kept a-rollin' all night long.
dave@howtogeek:~/work$
```

The functionality and sheer usefulness of `grep` definitely warrants you checking out [its man page](#).

15. groups

The `groups` command tells you which groups a user is a member of.

```
groups dave
groups mary
```

```
dave@howtogeek:~/work$ groups dave
dave : dave adm cdrom sudo dip plugdev lpadmin sambashare vboxsf geek
dave@howtogeek:~/work$
dave@howtogeek:~/work$ groups mary
mary : mary sudo
dave@howtogeek:~/work$
```

16. gzip

The `gzip` command compresses files. By default, it removes the original file and leaves you with the compressed version. To retain both the original and the compressed version, use the `-k` (keep) option.

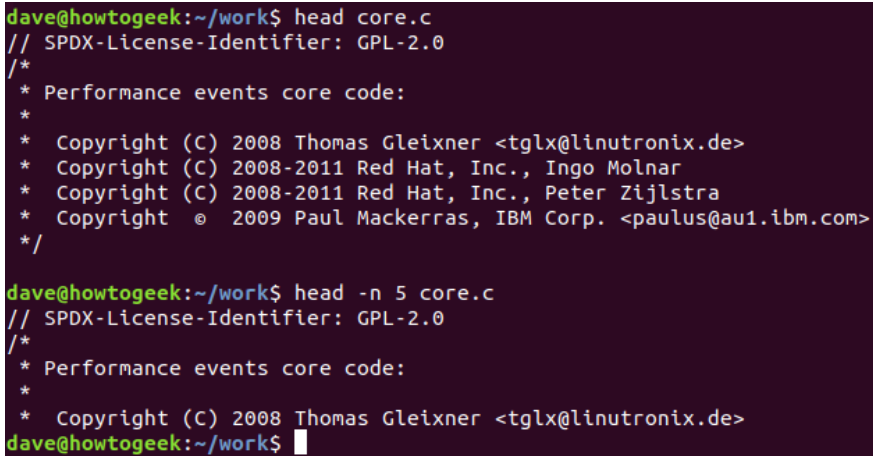
```
gzip -k core.c
```

```
dave@howtogeek:~/work$ gzip -k core.c
dave@howtogeek:~/work$ ls -l core*
-rw-r--r-- 1 dave dave 294075 Apr 24 14:33 core.c
-rw-r--r-- 1 dave dave  73416 Apr 24 14:33 core.c.gz
dave@howtogeek:~/work$
dave@howtogeek:~/work$
```

17. head

The `head` command gives you a listing of the first 10 lines of a file. If you want to see fewer or more lines, use the `-n` (number) option. In this example, we use `head` with its default of 10 lines. We then repeat the command asking for only five lines.

```
head -core.c
head -n 5 core.c
```

A terminal window with a dark purple background and a light blue border on the right. The prompt is 'dave@howtogeek:~/work\$'. The first command is 'head core.c', which outputs 10 lines of text: '// SPDX-License-Identifier: GPL-2.0', '/*', '* Performance events core code:', '*', '* Copyright (C) 2008 Thomas Gleixner <tglx@linutronix.de>', '* Copyright (C) 2008-2011 Red Hat, Inc., Ingo Molnar', '* Copyright (C) 2008-2011 Red Hat, Inc., Peter Zijlstra', '* Copyright © 2009 Paul Mackerras, IBM Corp. <paulus@au1.ibm.com>', and '*/'. The second command is 'head -n 5 core.c', which outputs the first five lines of the same text. The prompt returns to 'dave@howtogeek:~/work\$' with a cursor.

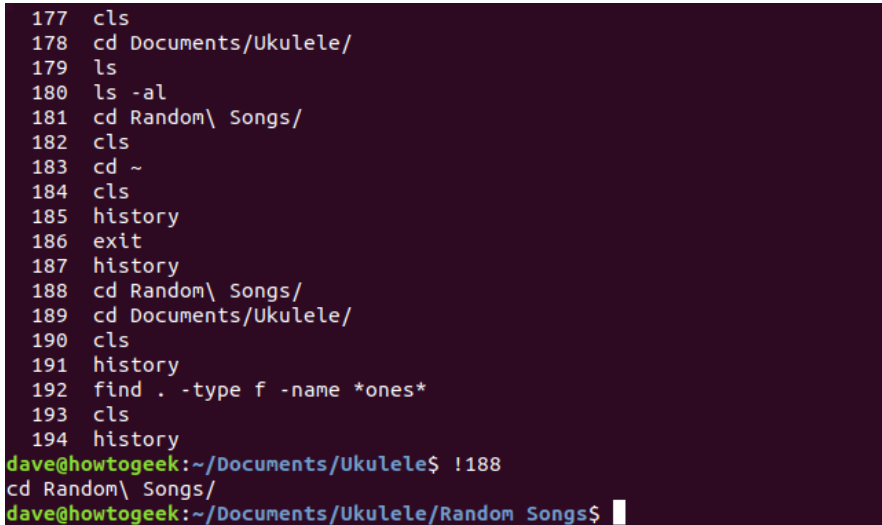
```
dave@howtogeek:~/work$ head core.c
// SPDX-License-Identifier: GPL-2.0
/*
 * Performance events core code:
 *
 * Copyright (C) 2008 Thomas Gleixner <tglx@linutronix.de>
 * Copyright (C) 2008-2011 Red Hat, Inc., Ingo Molnar
 * Copyright (C) 2008-2011 Red Hat, Inc., Peter Zijlstra
 * Copyright © 2009 Paul Mackerras, IBM Corp. <paulus@au1.ibm.com>
 */

dave@howtogeek:~/work$ head -n 5 core.c
// SPDX-License-Identifier: GPL-2.0
/*
 * Performance events core code:
 *
 * Copyright (C) 2008 Thomas Gleixner <tglx@linutronix.de>
dave@howtogeek:~/work$
```

18. history

The `history` command lists the commands you have previously issued on the command line. You can repeat any of the commands from your history by typing an exclamation point (!) and the number of the command from the history list.

```
!188
```

A terminal window with a dark purple background and light green text. It shows a sequence of commands: 177 cls, 178 cd Documents/Ukulele/, 179 ls, 180 ls -al, 181 cd Random\ Songs/, 182 cls, 183 cd ~, 184 cls, 185 history, 186 exit, 187 history, 188 cd Random\ Songs/, 189 cd Documents/Ukulele/, 190 cls, 191 history, 192 find . -type f -name *ones*, 193 cls, 194 history. Below the list, the prompt dave@howtogeek:~/Documents/Ukulele\$ is followed by !188, which repeats the command from line 188 (cd Random\ Songs/). The next prompt shows the user has moved to the directory: dave@howtogeek:~/Documents/Ukulele/Random Songs\$.

Typing two exclamation points repeats your previous command.

```
!!
```

19. kill

The `kill` command allows you to terminate a process from the command line. You do this by providing the process ID (PID) of the process to `kill`. Don't kill processes willy-nilly. You need to have a good reason to do so. In this example, we'll pretend the `shutter` program has locked up.

To find the PID of `shutter` we'll use our `ps` and `grep` trick from the section about the `alias` command, above. We can search for the `shutter` process and obtain its PID as follows:

```
ps -e | grep shutter.
```

Once we have determined the PID---1692 in this case---we can kill it as follows:

```
kill 1692
```

```
dave@howtogeek:~$ ps -e | grep shutter
1692 tty2      00:00:10  shutter
dave@howtogeek:~$ kill 1692
dave@howtogeek:~$ ps -e | grep shutter
dave@howtogeek:~$
```

20. less

The `less` command allows you to view files without opening an editor. It's faster to use, and there's no chance of you inadvertently modifying the file. With `less` you can scroll forward and backward through the file using the Up and Down Arrow keys, the PgUp and PgDn keys and the Home and End keys. Press the Q key to quit from `less`.

To view a file, provide its name to `less` as follows:

```
less core.c
```

```
// SPDX-License-Identifier: GPL-2.0
/*
 * Performance events core code:
 *
 * Copyright (C) 2008 Thomas Gleixner <tglx@linutronix.de>
 * Copyright (C) 2008-2011 Red Hat, Inc., Ingo Molnar
 * Copyright (C) 2008-2011 Red Hat, Inc., Peter Zijlstra
 * Copyright © 2009 Paul Mackerras, IBM Corp. <paulus@au1.ibm.com>
 */

#include <linux/fs.h>
#include <linux/mm.h>
#include <linux/cpu.h>
#include <linux/smp.h>
#include <linux/idr.h>
#include <linux/file.h>
#include <linux/poll.h>
#include <linux/slab.h>
#include <linux/hash.h>
#include <linux/tick.h>
core.c
```

You can also pipe the output from other commands into `less`. To see the output from `ls` for a listing of your entire hard drive, use the following command:

```
ls -R / | less
```

```
sbin
snap
srv
swapfile
sys
tmp
usr
var
vmlinuz
vmlinuz.old

/bin:
bash
brlty
bunzip2
busybox
bzip2
bzip2
bzdiff
bzegrep
:
```

Use `/` to search forward in the file and use `?` to search backward.

21. `ls`

This might be the first command most Linux users meet. It lists the files and folders in the directory you specify. By default, `ls` looks in the current directory. There are a great many options you can use with `ls`, and we strongly advise reviewing its [the man page](#). Some common examples are presented here.

To list the files and folders in the current directory:

```
ls
```

To list the files and folders in the current directory with a detailed listing use the `-l` (long) option:

```
ls -l
```

To use human-friendly file sizes include the `-h` (human) option:

```
ls -lh
```

To include hidden files, use the `-a` (all files) option:

```
ls -lha
```

```
dave@howtogeek:~/work$ ls
core.c  core.c.gz
dave@howtogeek:~/work$ ls -l
total 360
-rw-r--r-- 1 dave dave 294075 Apr 24 14:33 core.c
-rw-r--r-- 1 dave dave 73416 Apr 24 14:33 core.c.gz
dave@howtogeek:~/work$ ls -lh
total 360K
-rw-r--r-- 1 dave dave 288K Apr 24 14:33 core.c
-rw-r--r-- 1 dave dave 72K Apr 24 14:33 core.c.gz
dave@howtogeek:~/work$ ls -lha
total 372K
drwxr-xr-x  2 dave dave 4.0K Apr 24 14:45 .
drwxr-xr-x 22 dave dave 4.0K Apr 24 14:37 ..
-rw-r--r--  1 dave dave  55 Apr 24 14:45 .config
-rw-r--r--  1 dave dave 288K Apr 24 14:33 core.c
-rw-r--r--  1 dave dave  72K Apr 24 14:33 core.c.gz
dave@howtogeek:~/work$
```

22. man

The `man` command displays the "man pages" for a command in `less`. The man pages are the user manual for that command. Because `man` uses `less` to display the man pages, you can use the search capabilities of `less`.

For example, to see the man pages for `chown`, use the following command:

```
man chown
```

Use the Up and Down arrow or PgUp and PgDn keys to scroll through the document. Press `q` to quit the man page or press `h` for help.

```
CHMOD(1)                                User Commands                                CHMOD(1)

NAME
    chmod - change file mode bits

SYNOPSIS
    chmod [OPTION]... MODE[.MODE]... FILE...
    chmod [OPTION]... OCTAL-MODE FILE...
    chmod [OPTION]... --reference=RFIL FILE...

DESCRIPTION
    This manual page documents the GNU version of chmod. chmod
    changes the file mode bits of each given file according to
    mode, which can be either a symbolic representation of
    changes to make, or an octal number representing the bit pat-
    tern for the new mode bits.

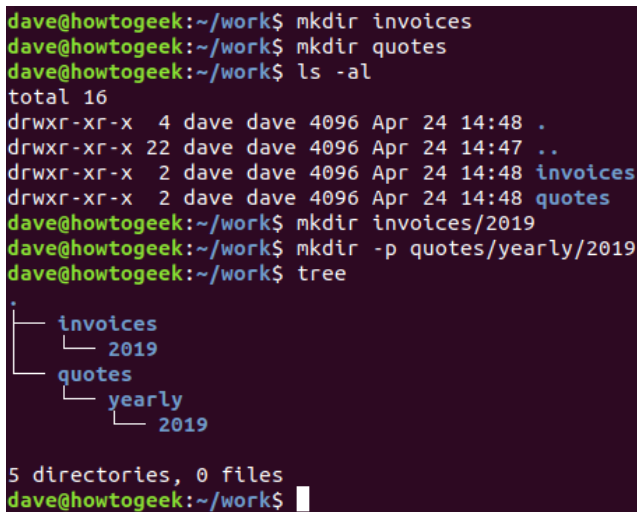
    The format of a symbolic mode is
    [ugoa...][[-+=[perms...]]...], where perms is either zero or
    more letters from the set rwXst, or a single letter from the
    Manual page chmod(1) line 1 (press h for help or q to quit)
```

23. mkdir

The `mkdir` command allows you to create new directories in the filesystem. You must provide the name of the new directory to `mkdir`. If the new directory is not going to be within the current directory, you must provide the path to the new directory.

To create two new directories in the current directory called "invoices" and "quotes," use these two commands:

```
mkdir invoices
mkdir quotes
```

A terminal window with a dark purple background and light green text. The user 'dave' is in the directory '~/work'. The commands and output are as follows:

```
dave@howtogeek:~/work$ mkdir invoices
dave@howtogeek:~/work$ mkdir quotes
dave@howtogeek:~/work$ ls -al
total 16
drwxr-xr-x  4 dave dave 4096 Apr 24 14:48 .
drwxr-xr-x 22 dave dave 4096 Apr 24 14:47 ..
drwxr-xr-x  2 dave dave 4096 Apr 24 14:48 invoices
drwxr-xr-x  2 dave dave 4096 Apr 24 14:48 quotes
dave@howtogeek:~/work$ mkdir invoices/2019
dave@howtogeek:~/work$ mkdir -p quotes/yearly/2019
dave@howtogeek:~/work$ tree
.
├── invoices
│   └── 2019
└── quotes
    ├── yearly
    │   └── 2019
```

At the bottom, it says '5 directories, 0 files' and the prompt 'dave@howtogeek:~/work\$' is shown with a cursor.

To create a new directory called "2019" inside the "invoices" directory, use this command:

```
mkdir invoices/2109
```

If you are going to create a directory, but its parent directory does not exist, you can use the `-p` (parents) option to have `mkdir` create all of the required parent directories too. In the following command, we are creating the "2019" directory inside the "yearly" directory inside the "quotes" directory. The "yearly" directory does not exist, but we can have `mkdir` create all the specified directories at once:

```
mkdir -p quotes/yearly/2019
```

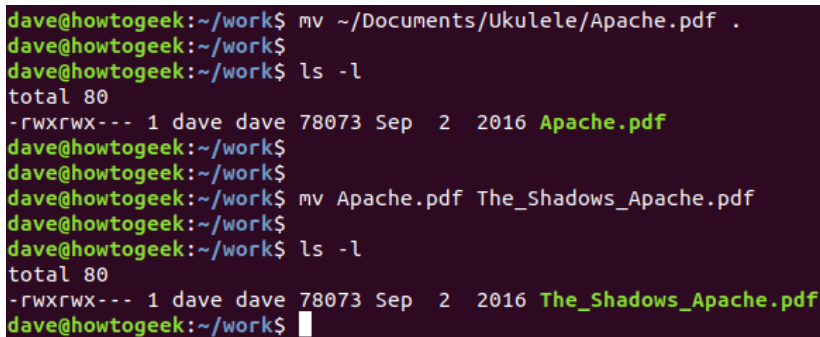
The "yearly" directory is also created.

24. mv

The `mv` command allows you to move files and directories from directory to directory. It also allows you to rename files.

To move a file you must tell `mv` where the file is and where you want it to be moved to. In this example, we're moving a file called `apache.pdf` from the "`~/Document/Ukulele`" directory and placing it in the current directory, represented by the single `.` character.

```
mv ~/Documents/Ukulele/Apache.pdf .
```

A terminal window with a dark purple background and green text. The user 'dave' is at the 'howtogeek' machine in the '~/.work' directory. They run 'mv ~/Documents/Ukulele/Apache.pdf .' to move a file. Then they run 'ls -l' and see a file 'Apache.pdf' with permissions '-rwxrwx---', size 78073, and date Sep 2 2016. They then run 'mv Apache.pdf The_Shadows_Apache.pdf' to rename it. Finally, they run 'ls -l' again and see the file 'The_Shadows_Apache.pdf' with the same permissions and size, confirming the move and rename were successful.

```
dave@howtogeek:~/work$ mv ~/Documents/Ukulele/Apache.pdf .
dave@howtogeek:~/work$
dave@howtogeek:~/work$ ls -l
total 80
-rwxrwx--- 1 dave dave 78073 Sep  2  2016 Apache.pdf
dave@howtogeek:~/work$
dave@howtogeek:~/work$ mv Apache.pdf The_Shadows_Apache.pdf
dave@howtogeek:~/work$
dave@howtogeek:~/work$ ls -l
total 80
-rwxrwx--- 1 dave dave 78073 Sep  2  2016 The_Shadows_Apache.pdf
dave@howtogeek:~/work$
```

To rename the file, you "move" it into a new file with the new name.

```
mv Apache.pdf The_Shadows_Apache.pdf
```

The file move and rename action could have been achieved in one step:

```
mv ~/Documents/Ukulele/Apache.pdf
./The_Shadows_Apache.pdf
```

25. passwd

The `passwd` command lets you change the password for a user. Just type `passwd` to change your own password.

You can also change the password of another user account, but you must use `sudo`. You will be asked to enter the new password twice.

```
sudo passwd mary
```

```
dave@howtogeek:~/work$ sudo passwd mary
[sudo] password for dave:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
dave@howtogeek:~/work$
```

26. ping

The `ping` command lets you verify that you have network connectivity with another network device. It is commonly used to help troubleshoot networking issues. To use `ping`, provide the IP address or machine name of the other device.

```
ping 192.168.4.18
```

The `ping` command will run until you stop it with `Ctrl+C`.

```
dave@howtogeek:~/work$ ping 192.168.4.18
PING 192.168.4.18 (192.168.4.18) 56(84) bytes of data.
64 bytes from 192.168.4.18: icmp_seq=1 ttl=64 time=1.49 ms
64 bytes from 192.168.4.18: icmp_seq=2 ttl=64 time=1.00 ms
64 bytes from 192.168.4.18: icmp_seq=3 ttl=64 time=0.947 ms
64 bytes from 192.168.4.18: icmp_seq=4 ttl=64 time=1.05 ms
64 bytes from 192.168.4.18: icmp_seq=5 ttl=64 time=0.995 ms
64 bytes from 192.168.4.18: icmp_seq=6 ttl=64 time=1.07 ms
64 bytes from 192.168.4.18: icmp_seq=7 ttl=64 time=1.07 ms
64 bytes from 192.168.4.18: icmp_seq=8 ttl=64 time=1.04 ms
64 bytes from 192.168.4.18: icmp_seq=9 ttl=64 time=1.03 ms
64 bytes from 192.168.4.18: icmp_seq=10 ttl=64 time=1.00 ms
64 bytes from 192.168.4.18: icmp_seq=11 ttl=64 time=1.06 ms
64 bytes from 192.168.4.18: icmp_seq=12 ttl=64 time=0.988 ms
64 bytes from 192.168.4.18: icmp_seq=13 ttl=64 time=1.01 ms
^C
--- 192.168.4.18 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12017ms
rtt min/avg/max/mdev = 0.947/1.060/1.490/0.135 ms
dave@howtogeek:~/work$
```

Here's what's going on here:

- The device at IP address 192.168.4.18 is responding to our ping requests and is sending back packets of 64 bytes.
- The [Internet Control Messaging Protocol](#) (ICMP) sequence numbering allows us to check for missed responses (dropped packets).
- The TTL figure is the "time to live" for a packet. Each time the packet goes through a router, it is (supposed to be) decremented by one. If it reaches zero the packet is thrown away. The aim of this is to prevent network loopback problems from flooding the network.
- The time value is the duration of the round trip from your computer to the device and back. Simply put, the lower this time, the better.

To ask `ping` to run for a specific number of ping attempts, use the `-c` (count) option.

```
ping -c 5 192.168.4.18
```

To hear a ping, use the `-a` (audible) option.

```
ping -a 192.168.4.18
```

27. `ps`

The `ps` command lists running processes. Using `ps` without any options causes it to list the processes running in the current shell.

```
ps
```



```
dave@howtogeek:~/work$ ps
  PID TTY          TIME CMD
  4214 pts/0    00:00:00 bash
  4609 pts/0    00:00:00 ps
dave@howtogeek:~/work$
```

To see all the processes related to a user, use the `-u` (user) option. This is likely to be a long list, so for convenience pipe it through `less`.

```
ps -u dave | less
```

```

PID TTY          TIME CMD
1123 ?             00:00:00 systemd
1124 ?             00:00:00 (sd-pam)
1137 ?             00:00:00 gnome-keyring-d
1141 tty2          00:00:00 gdm-x-session
1143 tty2          00:00:57 Xorg
1150 ?             00:00:00 dbus-daemon
1154 tty2          00:00:00 gnome-session-b
1254 ?             00:00:00 VBoxClient
1255 ?             00:00:00 VBoxClient
1264 ?             00:00:00 VBoxClient
1265 ?             00:00:00 VBoxClient
1269 ?             00:00:00 VBoxClient
1270 ?             00:00:00 VBoxClient
1274 ?             00:00:00 VBoxClient
1275 ?             00:00:41 VBoxClient
1286 ?             00:00:00 ssh-agent
1288 ?             00:00:00 at-spi-bus-laun
1293 ?             00:00:00 dbus-daemon
1296 ?             00:00:00 at-spi2-registr
:

```

To see every process that is running, use the `-e` (every process) option:

```
ps -e | less
```

28. pwd

Nice and simple, the `pwd` command prints the working directory (the current directory) from the root `/` directory.

```
pwd
```

```

dave@howtogeek:~/work$ pwd
/home/dave/work
dave@howtogeek:~/work$

```

29. shutdown

The `shutdown` command lets you [shut down or reboot your Linux system](#).


Using `shutdown` with no parameters will shut down your computer in one minute.

```
shutdown
```

```
dave@howtogeek:~/work$ shutdown
```

To shut down immediately, use the `now` parameter.

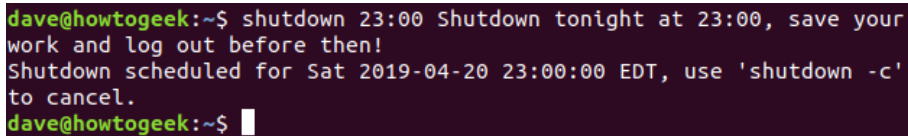
```
shutdown now
```



```
dave@howtogeek:~$ shutdown now
```

You can also schedule a shutdown and inform any logged in users of the pending shutdown. To let the `shutdown` command know when you want it to shut down, you provide it with a time. This can be a set number of minutes from now, such as `+90` or a precise time, like `23:00`. Any text message you provide is broadcast to logged in users.

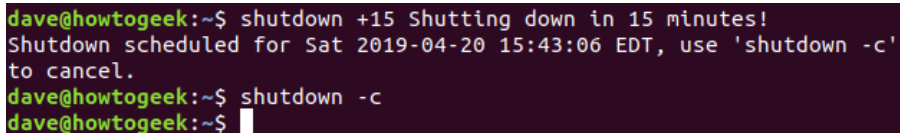
```
shutdown 23:00 Shutdown tonight at 23:00, save your work  
and log out before then!
```



```
dave@howtogeek:~$ shutdown 23:00 Shutdown tonight at 23:00, save your  
work and log out before then!  
Shutdown scheduled for Sat 2019-04-20 23:00:00 EDT, use 'shutdown -c'  
to cancel.  
dave@howtogeek:~$
```

To cancel a shutdown, use the `-c` (cancel) option. Here we have scheduled a shutdown for fifteen minutes time from now---and then changed our minds.

```
shutdown +15 Shutting down in 15 minutes!  
shutdown -c
```



```
dave@howtogeek:~$ shutdown +15 Shutting down in 15 minutes!  
Shutdown scheduled for Sat 2019-04-20 15:43:06 EDT, use 'shutdown -c'  
to cancel.  
dave@howtogeek:~$ shutdown -c  
dave@howtogeek:~$
```

30. SSH

Use the `ssh` command to make a connection to a remote Linux computer and log into your account. To make a connection, you must provide your user name and the IP address or domain name of the remote computer. In this example, the user `mary` is logging into the computer at `192.168.4.23`. Once the connection is established, she is asked for her password.

```
ssh mary@192.168.4.23
```

```
dave@Nostromo:~$ ssh mary@192.168.4.23
mary@192.168.4.23's password: 
```

Her username and password are verified and accepted, and she is logged in. Notice that her prompt has changed from "Nostromo" to "howtogeek."

Mary issues the `w` command to list the current users on "howtogeek" system. She is listed as being connected from `pts/1`, which is a pseudo-terminal slave. That is, it is not a terminal directly connected to the computer.

To close the session, mary types `exit` and is returned to the shell on the "Nostromo" computer.

```
w
exit
```

```
mary@howtogeek:~$ w
 15:02:34 up  4:24,  2 users,  load average: 0.17, 0.11, 0.09
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
dave      :0        :0               Tue15    ?xdm?  2:09   0.00s /usr/l
i
mary      pts/1     192.168.4.27     15:02    1.00s  0.05s  0.00s w
mary@howtogeek:~$ exit
logout
Connection to 192.168.4.23 closed.
dave@Nostromo:~$ 
```

31. sudo

The `sudo` command is required when performing actions that require root or superuser permissions, such as changing the password for another user.

```
sudo passwd mary
```

```
dave@howtogeek:~/work$ sudo passwd mary
[sudo] password for dave:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
dave@howtogeek:~/work$
```

32. tail

The `tail` command gives you a listing of the last 10 lines of a file. If you want to see fewer or more lines, use the `-n` (number) option. In this example, we use `tail` with its default of 10 lines. We then repeat the command asking for only five lines.

```
tail core.c
tail -n 5 core.c
```

```
dave@howtogeek:~/work$ tail core.c
    .attach          = perf_cgroup_attach,
    /*
     * Implicitly enable on dfl hierarchy so that perf events can
     * always be filtered by cgroup2 path as long as perf_event
     * controller is not mounted on a legacy hierarchy.
     */
    .implicit_on_dfl = true,
    .threaded        = true,
};
#endif /* CONFIG_CGROUP_PERF */
dave@howtogeek:~/work$ tail -n 5 core.c
    /*
     * Implicitly enable on dfl hierarchy so that perf events can
     * always be filtered by cgroup2 path as long as perf_event
     * controller is not mounted on a legacy hierarchy.
     */
    .implicit_on_dfl = true,
    .threaded        = true,
};
#endif /* CONFIG_CGROUP_PERF */
dave@howtogeek:~/work$
```

33. tar

With the `tar` command, you can create an archive file (also called a tarball) that can contain many other files. This makes it much more convenient to distribute a collection of files. You can also use `tar` to extract the files from an archive file. It is common to ask `tar` to compress the archive. If you do not ask for compression, the archive file is created uncompressed.

To create an archive file, you need to tell `tar` which files to include in the archive file, and the name you wish the archive file to have.

In this example, the user is going to archive all of the files in the Ukulele directory, which is in the current directory.

```
dave@howtogeek:~/work$ ls
Ukulele
dave@howtogeek:~/work$
```

They have used the `-c` (create) option and the `-v` (verbose) option. The verbose option gives some visual feedback by listing the files to the terminal window as they are added to the archive. The `-f` (filename) option is followed by the desired name of the archive. In this case, it is `songs.tar`.

```
tar -cvf songs.tar Ukulele/
```

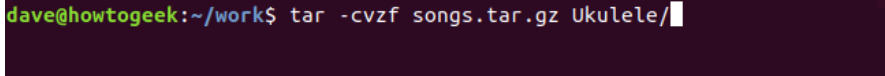
```
dave@howtogeek:~/work$ ls
Ukulele
dave@howtogeek:~/work$ tar -cvf songs.tar Ukulele/
```

The files are listed to the terminal window as they are added to the archive file.

There are two ways to tell `tar` that you want the archive file to be compressed. The first is with the `-z` (gzip) option. This tells `tar` to use the `gzip` utility to compress the archive once it has been created.

It is usual to add `".gz"` as suffix to this type of archive. That allows anyone who is extracting files from it to know which commands to pass to `tar` to correctly retrieve the files.


```
tar -cvzf songs.tar.gz Ukulele/
```

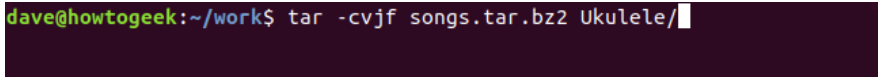


```
dave@howtogeek:~/work$ tar -cvzf songs.tar.gz Ukulele/
```

The files are listed to the terminal window as they are added to the archive file as before, but the creation of the archive will take a little longer because of the time required for the compression.

To create an archive file that is compressed using a superior compression algorithm giving a smaller archive file use the `-j` (bzip2) option.

```
tar -cvjf songs.tar.bz2 Ukulele/
```

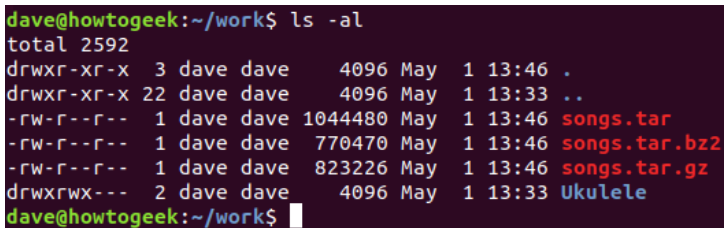


```
dave@howtogeek:~/work$ tar -cvjf songs.tar.bz2 Ukulele/
```

Once again, the files are listed as the archive is created. The `-j` option is noticeably slower than the `-z` option.

If you are archiving a great many files, you must choose between the `-z` option for decent compression and reasonable speed, or the `-j` option for better compression and slower speed.

As can be seen in the screenshot below, the ".tar" file is the largest, the ".tar.gz" is smaller, and the ".tar.bz2" is the smallest of the archives.



```
dave@howtogeek:~/work$ ls -al
total 2592
drwxr-xr-x  3 dave dave  4096 May  1 13:46 .
drwxr-xr-x 22 dave dave  4096 May  1 13:33 ..
-rw-r--r--  1 dave dave 1044480 May  1 13:46 songs.tar
-rw-r--r--  1 dave dave  770470 May  1 13:46 songs.tar.bz2
-rw-r--r--  1 dave dave  823226 May  1 13:46 songs.tar.gz
drwxrwx---  2 dave dave  4096 May  1 13:33 Ukulele
dave@howtogeek:~/work$
```

To extract files from an archive file, use the `-x` (extract) option. The `-v` (verbose) and `-f` (filename) options behave as they do when creating archives. Use `ls` to confirm which type of archive you are going to extract the files from, then issue the following command:

```
ls
tar -xvf songs.tar
```

```
dave@howtogeek:~/work$ ls
songs.tar  songs.tar.bz2  songs.tar.gz
dave@howtogeek:~/work$ tar -xvf songs.tar
```

The files are listed as they are extracted. Note that the Ukulele directory is also recreated for you.

To extract files from a ".tar.gz" archive, use the `-z` (gzip) option.

```
tar -xvzf songs.tar.gz
```

```
dave@howtogeek:~/work$ tar -xvzf songs.tar.gz
```

Finally, to extract files from a ".tar.bz2" archive use the `-j` option instead of the `-z` (gzip) option.

```
tar -xvjf songs.tar.bz2
```

```
dave@howtogeek:~/work$ tar -xvjf songs.tar.bz2
```

34. top

The `top` command shows you a real-time display of the data relating to your Linux machine. The top of the screen is a status summary.

The first line shows you the time and how long your computer has been running for, how many users are logged into it, and what the load average has been over the past one, five, and fifteen minutes.

The second line shows the number of tasks and their states: running, stopped, sleeping and zombie.

The third line shows CPU information. Here's what the fields mean:

- `us`: value is the CPU time the CPU spends executing processes for users, in "user space"
- `sy`: value is the CPU time spent on running system "kernel space" processes
- `ni`: value is the CPU time spent on executing processes with a manually set nice value
- `id`: is the amount of CPU idle time
- `wa`: value is the time the CPU spends waiting for I/O to complete

- hi: The CPU time spent servicing hardware interrupts
- si: The CPU time spent servicing software interrupts
- st: The CPU time lost due to running virtual machines ("steal time")

The fourth line shows the total amount of physical memory, and how much is free, used and buffered or cached.

The fifth line shows the total amount of swap memory, and how much is free, used and available (accounting for memory that is expected to be recoverable from caches).

```
top - 15:56:38 up 5:18, 1 user, load average: 0.16, 0.07, 0.02
Tasks: 223 total, 1 running, 190 sleeping, 0 stopped, 0 zombie
%Cpu(s): 9.2 us, 4.0 sy, 0.0 ni, 86.8 id, 0.0 wa, 0.0 hi, 0.0 si
KiB Mem : 2041112 total, 77240 free, 1161664 used, 802208 buff/
KiB Swap: 483800 total, 424652 free, 59148 used. 191124 avail
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
1314	dave	20	0	2958.6m	276.7m	75.7m	S	11.6	13.9	4:11.16
1143	dave	20	0	989.8m	569.3m	497.5m	S	3.6	28.6	1:03.56
4289	dave	20	0	771.3m	119.2m	36.9m	S	2.0	6.0	0:05.70
4204	dave	20	0	845.1m	35.9m	26.7m	S	0.7	1.8	0:12.91
1275	dave	20	0	122.8m	1.6m	1.2m	S	0.3	0.1	0:51.23
1345	dave	20	0	353.6m	8.5m	6.2m	S	0.3	0.4	0:21.69
1539	dave	20	0	807.0m	59.9m	46.7m	S	0.3	3.0	0:03.69
4970	dave	20	0	47.7m	3.6m	3.0m	R	0.3	0.2	0:00.17
1	root	20	0	156.1m	8.4m	6.4m	S	0.0	0.4	0:03.30
2	root	20	0	0.0m	0.0m	0.0m	S	0.0	0.0	0:00.00
3	root	0	-20	0.0m	0.0m	0.0m	I	0.0	0.0	0:00.00
4	root	0	-20	0.0m	0.0m	0.0m	I	0.0	0.0	0:00.00
6	root	0	-20	0.0m	0.0m	0.0m	I	0.0	0.0	0:00.00
8	root	0	-20	0.0m	0.0m	0.0m	I	0.0	0.0	0:00.00

The user has pressed the E key to change the display into more humanly digestible figures instead of long integers representing bytes.

The columns in the main display are made up of:

- PID: Process ID
- USER: Name of the owner of the process
- PR: [Process priority](#)
- NI: The nice value of the process
- VIRT: Virtual memory used by the process
- RES: Resident memory used by the process
- SHR: Shared memory used by the process
- S: Status of the process. See the list below of the values this field can take
- %CPU: the share of CPU time used by the process since last update
- %MEM: share of physical memory used
- TIME+: total CPU time used by the task in hundredths of a second

- **COMMAND:** command name or command line (name + options)

(The command column didn't fit into the screenshot.)

The status of the process can be one of:

- D: Uninterruptible sleep
- R: Running
- S: Sleeping
- T: Traced (stopped)
- Z: Zombie

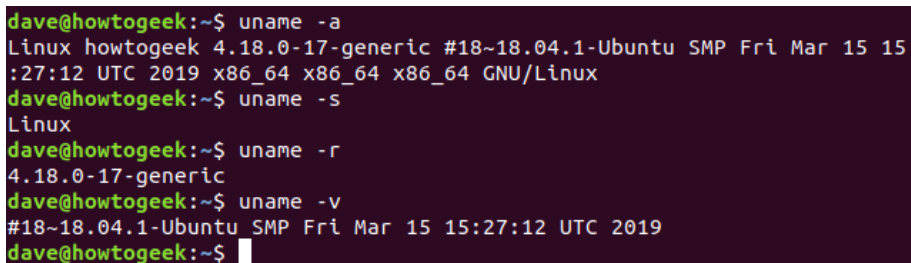
Press the Q key to exit from `top`.

35. `uname`

You can obtain some system information regarding the Linux computer you're working on with the `uname` command.

- Use the `-a` (all) option to see everything.
- Use the `-s` (kernel name) option to see the type of kernel.
- Use the `-r` (kernel release) option to see the kernel release.
- Use the `-v` (kernel version) option to see the kernel version.

```
uname -a
uname -s
uname -r
uname -v
```

A terminal window with a dark purple background and green text. It shows the execution of the 'uname' command with various flags. The output for 'uname -a' is 'Linux howtogeek 4.18.0-17-generic #18~18.04.1-Ubuntu SMP Fri Mar 15 15:27:12 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux'. The output for 'uname -s' is 'Linux'. The output for 'uname -r' is '4.18.0-17-generic'. The output for 'uname -v' is '#18~18.04.1-Ubuntu SMP Fri Mar 15 15:27:12 UTC 2019'. The prompt 'dave@howtogeek:~\$' is visible at the start of each line.

```
dave@howtogeek:~$ uname -a
Linux howtogeek 4.18.0-17-generic #18~18.04.1-Ubuntu SMP Fri Mar 15 15
:27:12 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
dave@howtogeek:~$ uname -s
Linux
dave@howtogeek:~$ uname -r
4.18.0-17-generic
dave@howtogeek:~$ uname -v
#18~18.04.1-Ubuntu SMP Fri Mar 15 15:27:12 UTC 2019
dave@howtogeek:~$
```

36. w

The `w` command lists the currently logged in users.

`w`

```
dave@howtogeek:~/Documents$ w
16:02:09 up 5:24, 4 users, load average: 0.23, 0.12, 0.03
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
dave      :0        :0               Tue15    ?xdm?  2:34   0.00s /usr/l
t
tom       pts/1    192.168.4.25    15:59    2:28   0.03s   0.03s -bash
mary      pts/2    192.168.4.27    16:00    2:08   0.03s   0.03s -bash
dick      pts/3    192.168.4.27    16:00    27.00s  0.04s   0.04s -bash
dave@howtogeek:~/Documents$
```

37. whoami

Use `whoami` to find out who you are logged in as or who is logged into an unmanned Linux terminal.

`whoami`

```
dave@howtogeek:~/Documents$ whoami
dave
dave@howtogeek:~/Documents$
```

That's Your Toolkit

Learning Linux is like learning anything else. You're going to need some practice before become familiar with these commands. Once you have these commands at your fingertips, you'll be well along the path to proficiency.

There's an old joke---probably as old as [Unix](#) itself---that says the only command you need to know is the `man` command. There's a glimmer of truth in that, but some of the man pages are impenetrable without an introduction. This tutorial should give you the introduction you need.