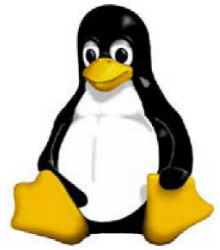
Modified from Source: Learn Code the Hard Way by Zed A. Shaw
BASH Specific Commands: BASH Features

Additional Source: LINUX Newbie Administrator Guide
Simon Sheppard's A- Z BASH Command Reference

Linux Bash Shell Cheat Sheet

(works with almost every distribution, except for apt-get which is Ubuntu/Debian exclusive)

Legend:



… = etc., more than one file can be affected

<ctrl-d> = keystroke simultaneous, i.e. "Hold the control key and the 'd' key down simultaneously, but don't type the '-' "

<s><d> = keystroke sequence, i.e. "Press the 's' key, release it, then press the 'd' key and release it"

{*filename*} or {*foldername*}, etc. = replacement, i.e. replace 'filename' with the name of the file with which you are working

Do not include the equal sign used below in your actual commands!

Basic Commands

Basic Terminal Shortcuts

<ctrl-a> = Cursor to start of line

<ctrl-c> = Kill the current process

<ctrl-d> = Logout

<ctrl-e> = Cursor the end of line

<ctrl-k> = Delete right of the cursor

<ctrl-l> = Clear the terminal (that's an 'L')

<ctrl-r> = reverse search history

<ctrl-u> = Delete left of the cursor

<ctrl-w> = Delete word on the left

<ctrl-y> = Paste

<ctrl-z> = stops the current command

<!><!> = repeat last command

<shft><Page Up> / <shft><Page Down> = Go up/down the terminal

<tab> = auto completion of file or command

> = redirect the output of a command to a file; will overwrite

>> = redirect the output of a command to a file; will append

Basic file manipulation

. = this directory

.. = this directory plus one more, i.e. 'cd ..'

If a filename or foldername has spaces, enclose the name with double quotes, i.e. "file name"

cat {*filename*} = show content of file
optional: less, more i.e. "cat {*filename*} less",
are used to paginate the output

cp {*filename1*} {*filename2*} = copy and rename a file

cp {*filename*} {*foldername*}/ = copy to folder
(make sure to include the trailing slash /)

cp {*filename*} {*foldername*}/{*newfilename*} = copy to an existing folder with a new file name

cp -R {*foldername1*} {*foldername2*} = copy and rename a folder
'-R' means 'recursive' which includes all of the folder contents with the new copy

cp {**.fileextension*} {*foldername*}/ = copy all of '.file type' to a folder
i.e. *.txt is all files with the extension '.txt'
(make sure to include the trailing slash /)

head {*filename}* = from the top
optional: -n
head -n {number of lines from the top} {*filename*}

ln {*filename1*} {*filename2*} = create a physical link

ln -s {*filename1*} {*filename2*} = create a symbolic link

mkdir {*foldername*} = create a directory
i.e. mkdir {*myStuff*} …
i.e. mkdir {*myStuff/pictures/*} …

mv {*filename*} {*foldername*}/ = move file to a folder
(make sure to include the trailing slash /)

mv {*foldername1*} {*foldername2*} = move folder to a new name inside the original folder where it was located

mv {*filename1*} {*filename2*} = rename file

mv {*filename*} {*foldername*}/{*newfilename*} = move file to an existing folder with a new file name

mv {*foldername*}/ … = move folder up one level in hierarchy

rm {*filename*} … = delete file(s)

rm -i {*filename*} … = ask for confirmation each file

rm -f {*filename*} = force deletion of a file

rm -r {*foldername*}/ = delete folder

tail {*filename}* = from the bottom
optional: -n
tail -n {*number of lines from the bottom*} {*filename*}

touch {*filename*} = create or update a file

Commands for Compressed Files

Use just like the uncompressed command without the 'z'

zcat, zgrep, zdiff, zcmp, zmore, zless

Basic Terminal Navigation

If a filename or foldername has spaces, enclose the name with double quotes, i.e. "file name"

ls -a = list all files and folders

ls {*foldername*} = list files in folder

ls -lh = detailed list, human readable

ls -l {*.*fileextension*} = list all files with that extension

ls -lh {*filename*} = result for the named file only, human readable

cd {*foldername*} = change directory; if the folder name has spaces enclose the name with double quotes, i.e. "folder name"

cd / = go to the root of that file system

cd .. = go up one folder

cd ../../ **…** = go up two (or more) folders

pwd = print working directory; sometimes called "present working directory", but print is more accurate as this command prints the current directory to stdout, which is normally the screen

Researching Files

locate {text} = search the content of all files for {text}

locate {*filename*} = search for a file named {*filename*}

sudo updatedb = update database of files; must install the mlocate or slocate packages to get "updatedb"

find -name "{*filename*}" = search for a specific filename

find -name "{*filename**}" = search for one or more files that begin with the specified name

find -name "{**filename*}" = search for one or more files that end with the specified name

Advanced Search:

The 'find' command is the faster option, and is very powerful. Please refer to http://ss64.com/bash/find.html for more information. You can also use the 'man find' command to see the standard help file.

Create and Modify User Accounts

sudo adduser {*newusername*} = root creates new user

sudo passwd {*useraccountname*} = change a user's password

sudo deluser {*useraccountname*} = delete a user's account

groupadd {*newgroupname*} = create a new user group

groupdel {*groupname*} = delete a user group

usermod -g {*groupname*} {*useraccountname*} = add user to a group

usermod -g {*useraccountname*} {*newuseraccountname*} = change a user account name

usermod -aG {*groupname*} {*useraccountname*} = add groups to a user without losing their existing group membership

Get Help

man <*command*> = shows help for that command (RTFM)
man = manual; RTFM = Read The (Fine) Manual

Extract, Sort and Filter Data

The 'grep' command has many options, and is very powerful. Please refer to http://ss64.com/bash/grep.html for more information. You can also use the 'man grep' command to see the standard help file.

grep {*sometext*} {*filename*} = search for text in file
-i = case-*in*sensitive
-I = exclude binary files (upper case 'i')

grep -r {*sometext*} {*foldername*}/ = search for file names with the occurrence of {*sometext*} in {*foldername*}

With regular expressions:

grep -E ^{*sometext*} {*filename*} = search for the occurrence of {*sometext*} at the start of lines inside {*somefile*}

grep -E <0-4> {*filename*} = shows the lines in {*filename*} which contain the numbers 0-4

grep -E <a-zA-Z> {*filename*} = retrieve all lines in {*filename*} with alphabetical letters

sort {*filename*} = sort the lines in {*filename*} alphabetically

sort -o {*filename*} {*outputfilename*} = sort the lines in {*filename*}, then write result to {*outputfilename*}

sort -r {*filename*} = sort the lines in {*filename*} in reverse

sort -R {*filename*} = sort the lines in {*filename*} randomly

sort -n {*filename*} = sort a series of numbers in {*filename*}

wc {*filename*} = outputs the number of lines, number of words, and the byte size for {*filename*}; wc stands for "word count"
-l (lines), -w (words), -c (byte size), -m (number of characters)

cut –c 2-5 {*filename*} = cut the characters 2 to 5 of each line in {*filename*}
-d (delimiter) (-d & -f good for .csv files)
-f (# of field to cut)

Time Settings

The 'date' command allows you to view & modify the time on your computer

View: For Viewing the System Time

date "+%H" = shows the hour in 24-hour format, i.e. if it's 9 AM, then it will show 09. If it's 9 PM, then it will show 21.

date “+%H:%M:%S” = shows the hour in 24-hour format, the minutes and seconds of a date, i.e. if it's 9:02 AM, then it will show 09:02:00. If it's 9 PM, then it will show 21:02:00.

date "+%Y" = shows the year in 4-digit format, i.e. 2015

Modify: For Modifying the System Time

ex: MMDDhhmmYYYY = Month | Day | Hours | Minutes | Year

sudo date 031423421997 = March 14th 1997, 23:42

Chaining Commands

| = the pipe command; redirects the output of one command into another *command*, i.e. 'du | sort -nr | less

Process Modification

who = who is logged on and what they are doing

-d --{*delay*}= delay between updates in 1/10 seconds
-s --sort-key {*column*} = sort by {*column*}
-u --user={*useraccountname*} = process's from given user

top = Dynamic process list; quit top using <ctrl-c>
-d {delay} = delay between updates in 1/10 second
-i = suppress display of idle and zombie processes
-p {*PID*} = only monitor process with process ID {PID}

kill {PID} = kill a process with process ID {PID}

To get the PID # of the process use ps.

The 'ps' command is very powerful. Please refer to http://ss64.com/bash/ps.html for more information. You can also use the 'man ps' command to see the standard help file.

ps -u {*useraccountname*} | grep {*application*} = get the process id for application {*application*} run under user account name {*useraccountname*}

kill -9 {PID} = kill with extreme prejudice

kill -15 {PID} = kill with orderly shutdown

killall {*processname*} = stop multiple processes by name instead of process ID, i.e. 'killall -9 mozilla-bin'

sudo halt = stops all processes in an orderly fashion, but doesn't power off the computer

sudo reboot = reboots the computer in an orderly fashion

File Permissions

chown {*useraccountname*} {*filename*} = change the owner of file {*filename*}, i.e. 'chown bob hello.txt'

chown {*user*}:{*group*} {*filename*} = changes the ownership of file name {*filename*} to user {*user*} and group {*group*}

chown -R {*user*}:{*group*} {*foldername*}/{*filename*} = recursively changes the ownership of file name {*filename*} and the folder name {*foldername}* to user {*user*} and group {*group*}

chmod = modify user access/permission
u = user who owns the file or director
g = group which owns the file or directory
o = other; all users and groups which don't own the file or directory
a = all users; effectively u+g+o
r = read (read permissions)
w = write (write permissions)
x = execute (only useful for scripts and programs)
X = execute only if the target is a directory
'+' = add a permission
'-' = delete a permission
'=' = only affect the permissions that the file already has
read = list files in a directory
write = add new files to a directory or over-write a file
execute = access or execute (not just read) a file

chmod a-x {*filename*} = deny execute permissions for everyone

chmod a+r {*filename*} = add read access for everyone

The 'chmod' command has many options, and is very powerful. Please refer to http://ss64.com/bash/chmod.html for more information. You can also use the 'man chmod' command to see the standard help file.

Flow Redirection

> = redirect the output of a command to a *file*; will overwrite; i.e. 'who >userActivity.txt'

>> = redirect the output of a command to a *file*; will append

Redirect Errors:

Input/output streams can be referenced by numbers:

0 = stdin; standard input
1 = stdout; standard output
2 = stderr; standard error

Use this with redirection by placing the I/O stream number before the redirection operator (>); i.e. 'grep -E <0-4> {*filename*} 2>grepErr.txt in order to redirect the errors from the grep search into a text file

2>&1 = redirect errors to the standard output,
i.e. 'find /usr/home -name .profile 2>&1 | less'

Read progressively from the keyboard

{Command} << {wordToTerminateInput} = i.e. sort << END; where the word that terminates the input can be any word you choose

Input from keyboard where '>' is the terminal prompt:

> Hello
> Alex
> Cinema
> Game
> Code
> Ubuntu
> END

Terminal output:
Hello
Alex
Cinema
Game
Code
Ubuntu

Archive and Compress Data the Slow Way

The 'tar', 'gzip' and 'bzip2' command have many options, and are very powerful. Please refer to The Geek Stuff Tar Reference for more information.

1. Put {*filename1*}, {*filename2*}, {*filename3*} into the same {*foldername*}/, i.e. 'mv *.txt textfiles/'

2. Create the tar file, often called a tarball, which is not compressed

   tar {*archivefilename*}.tar {*foldername*}/
   -c = creates a .tar archive
   -v = tells you what is happening (verbose)
   -f = assembles the archive into one file, in this case named {*archivefilename*}.tar

   i.e. 'tar -cvf archivetext.tar textfiles/'

3. Create the gzip file, which compressed the archive tarball

   gzip {*archivefilename*}.tar

   i.e. 'cd textfiles/' then 'gzip archivetext.tar'

4. Optionally, use bzip2, which is slower but more powerful

   bzip2 {*archivefilename*}.tar

   i.e. 'cd textfiles/' then 'bzip2 archivetext.tar'

Data Decompression

gunzip {*archivefilename*}.tar.gz = decompress '{*archivefilename*}.tar.gz'

bunzip2 {*archivefilename*}.tar.bz2 = decompress '{*archivefilename*}.tar.bz2'

Data Unarchiving

tar -xvf {*archivefilename*}.tar = unarchive '{*archivefilename*}.tar'
-x = extracts a .tar archive
-v = tells you what is happening (verbose)
-f = perform the operation on the filename listed, in this case {*archivefilename*}.tar

i.e. 'tar -xvf archivetext.tar'

Archive and Compress Data the Fast Way:

While tar cannot compress files, only archive them, it can use either gzip or bzip2 to compress or decompress files and then archive or unarchive them in a single step. The process is the same, you just use a different switch (z or j) to specify preferences. The switches are tar switches, not gzip or bzip2 switches, so keep the meanings straight!

Tar with Gzip Compression

tar -zcvf {*archivefilename*}.tar.gz {*foldername*}/ = compress and archive the files in folder {*foldername*}/ to the compressed file {*archivefilename*}.tar.gz
-z = compress the file using gzip
-c = creates a .tar archive
-v = tells you what is happening (verbose)
-f = assembles the archive into one file, in this case named {*archivefilename*}.tar.gz

Tar with Gzip Decompression

tar -zxvf {*archivefilename*}.tar.gz {*foldername*}/ = decompress and unarchive the files in {*archivefilename*}.tar.gz into folder {*foldername*}
-z = decompress the file using gzip
-x = extract the .tar archive
-v = tells you what is happening (verbose)
-f = perform the operation on the filename listed, in this case {*archivefilename*}.tar.gz

Tar with Bzip2 Compression

tar -jcvf {*archivefilename*}.tar.bz2 {*foldername*}/ = compress and archive the files in folder {*foldername*}/ to the compressed file {*archivefilename*}.tar.bz2
-j = compress the file using bzip2
-c = creates a .tar archive
-v = tells you what is happening (verbose)
-f = assembles the archive into one file, in this case named {*archivefilename*}.tar.bz2

Tar with Bzip2 Decompression

tar -zxvf {*archivefilename*}.tar.bz2 {*foldername*}/ = decompress and unarchive the files in {*archivefilename*}.tar.bz2 into folder {*foldername*}
-j = decompress the file using bzip2
-x = extract the .tar archive
-v = tells you what is happening (verbose)
-f = perform the operation on the filename listed, in this case {*archivefilename*}.tar.bz2

Show the Contents of .tar, .gz or .bz2 Files without Decompressing

In the section above, we saw that there are a lot of different ways to archive/unarchive and/or compress/decompress files. Please refer to that section for a more verbose explanation of the following processes.

tar -tvf {*archivefilename*}.tar = list the files contained in filename {*archivefilename*}.tar

-t = list the files in the archive
-v = tells you what is happening (verbose)
-f = perform the operation on the filename listed, in this case the tarball {*archivefilename*}.tar

Commands for Compressed Files

Follow the links provided to find out more information on how to use these utilities. Basically, you use them just like the uncompressed command without the 'z'

zcat, zgrep, zdiff, zcmp, zmore, zless

Installing Software when it is Available in the Repositories

Using APT, the Advanced Package Manager

sudo apt-get install {softwarename} = perform this command as the privileged user, using APT to GET the package named {softwarename}

i.e. sudo apt-get install build-essential

Installing Software

The very best thing you can do when installing software is to read the INSTALL or README files which should tell you how the author expects them to be installed. To do that, see the entry for 'cat' in this file.

Installing Software When You "Compile From Source" After Downloading a Compressed Program From the Internet

Create a folder in which to place the file, i.e.

1. 'mkdir /home/username/src', which may already exist.
2. Move the file to that folder, i.e. 'mv {*filename*} /home/username/src/'
3. CD to that folder, i.e. 'cd /home/username/src'
4. Verify that the file is there, i.e. 'ls {*filename*}'
5. Decompress the file using the instructions from those sections. You should see a new directory when you are done.
6. CD to that new directory
7. Verify there is an INSTALL or README file, i.e. 'ls INSTALL' or 'ls README'
8. Read the INSTALL or README file, i.e. 'cat README' or 'less INSTALL' and follow the author's instructions. They may look something like the instructions below.

Creating a Makefile

It's very possible that when you create the makefile below, you will find that there are required packages missing from your system. Be sure to read the errors and install those programs as well.

Look for a file named 'configure'', i.e. 'ls configure'

If it exists, execute the configure file, i.e. ' ./configure', which checks your system for dependencies and creates a makefile.

Build the application binaries by running make, i.e. 'make'. If there are no errors, your program is ready to install.

Install the program, i.e. 'sudo make install'

Read the README file if you haven't already done so, i.e. 'less README'