Basic Linux commands by example

This is intended to be a quick reference to some of the most useful Linux commands.

All these commands have many options to make them more versatile. Do "man *command*" to see the full description of what can be done with these commands.

AREA	FUNCTION	COMMAND	ACTION
Directories	change	cd	Change to my home directory
		cd ..	Go up one level
		cd /dir1/dir2	Change to this absolute directory
		cd dira/dirb	Change to directory relative to current directory
	create	mkdir mydir	Make new directory here
	delete	rmdir mydir	Remove directory (must be empty)
	display	pwd	Print working directory
	paths	basename a/b/file.txt	Extract filename from path: "file.txt"
		dirname a/b/file.txt	Extract directories from path: "a/b"
	permissions	r	You may LIST FILES in that directory
		w	You may CREATE AND DELETE FILES in that directory
		x	You may cd to that directory
	shortcuts	.	Current directory
		..	Parent directory
		~	Home directory
Files	append	echo "good bye" >> x.txt	Append text to end of file
		cat x.txt >> y.txt	Append first file to end of second file
	compare	diff file1 file2	Compare two text files
		diff3 file1 file2 file3	Compare three files
		bdiff file1 file2	Compare very large files
		cmp file1 file2	Compare files byte by byte
	concatenate	cat myfile1 myfile2 > myfile3	Concatenate two files, making a new file.
	copy	cp file1 file2	Copy file1 to file2
	count	wc myfile	Count lines, words, and characters in file.
	create	echo "hello" > myfile	Create new file with this text
		echo -n > myfile	Create a zero length file
		touch myfile	Create a zero length file (if it doesn't already exist)
		vi x.txt	Create new file with editor (see below)

AREA	FUNCTION	COMMAND	ACTION
...files	delete	rm x.txt	Remove a file. You don't have to be the owner of the file, but you must have w (write) permission on the directory containing the file.
		rm -r dir1	Remove dir1 and everything in it
		rm -rf dir1/*	Clear out everything under dir1, but keep dir1 The “f” handles the case where there happens to already be nothing under dir1.
		rm dir1	DO NOT USE dir1 is a directory This will ALWAYS fail because rm by itself will not remove a directory
		rm dir1/*	DO NOT USE Remove all files in dir1, but not dir1 itself This will fail if there any subdirectories
		rm -r dir1/*	DO NOT USE Remove everything under dir1, but not dir1 itself This will fail if there is nothing under dir1.
	display	cat myfile	Display whole file on terminal
		head myfile	Show first 10 lines
		tail myfile	Show last 10 lines
		less myfile	Page file on terminal. (space to advance, "q" to quit)
		od -Ad -tu1 myfile | more	Show decimal values of each byte of the file, paging output
	filter	tr -d "\015" < file1 > file2	Remove all carriage returns from a file (CRLF > LF)
		tr -s " " " " < file1 > file2	Compress consecutive blanks to one blank
	find	find . -name myfile -type f -print	Find file myfile searching from current directory
		find / -name mydir -type d -print	Find directory mydir searching entire hierarchy (slow!)
		find . -name 'ufw*' -type f -print	Find matching files, note quotes to stop shell expansion
	list	ls	List files in current directory.
		ls -l *or* ll	Detailed (long) list.
		ls -a	Show hidden files (they start with '.')
	move	mv tax.txt mydir/	Move tax.txt file to directory mydir
		mv tax.txt /arch/2004/	Copy tax.txt to different volume
			So mv renames, moves, or copies files depending on the target. rename - if the source and target directories are the same move - if the source and target directories are on the same logical volume copy - if the source and target directories are on different logical volumes
	ownership	chown frank:sales x.txt	Change ownership of x.txt to user frank and group sales. Only the owner change ownership to someone else.
		chgrp sales y.txt	Just change owning group to sales

AREA	FUNCTION	COMMAND	ACTION
...files	permissions	r	You may read that file
		w	You may write to that file
		x	You may execute that file
		chmod 700 myfile	
		umask 0000	Set default create file permissions to 777 (open to world).
	rename	mv sales.txt account.txt	Rename sales.txt to account.txt
		rename ‘s/\.txt$/\.dat/’ *.txt	Rename all .txt files to .dat
			(use -v to see, -n to do dry run)
		rename ‘y/A-Z/a-z/’ FRED*.*	Change all FRED*.* filenames to lower case
		rename 's/(^..)\.png/c$1\.png/' *.png	Insert a ‘c’ at the front of all filenames nn.png Note $1 syntax, usual is \1 to refer to preceding group 1
	search	grep abc myfile	Search myfile for occurrences of "abc"
	sort/merge/extract	sort file1 > file2	Sort file on whole line, creating new output file.
		comm -1 x.txt y.txt > z.txt	Compare sorted files and suppress lines unique to x.txt
		cut -b 10-20,35-40 x.txt > y.txt	Put columns 10-20 and 35-40 into a new file
		join -o 1.3 2.2 x.txt y.txt > z.txt	Merge x.txt and y.txt on common field and output column 3 from x.txt and column2 from y.txt
		paste x.txt y.txt > z.txt	Paste lines together, separated by tabs
		paste -d "\n" a.txt b.txt c.txt > z.txt	Splice files together line by line.
		uniq myfile > newfile	Remove adjacent repeated lines
		uniq -f1 -u myfile > newfile	Ignore first field when testing for duplicate lines, and print just the lines that don't have duplicates
	split	split -l 1000 myfile	Split myfile into pieces xaa, xab, xac ... of 1000 lines each
	type	file xyz	What kind of file is this?
	wildcards	ls abc?.txt	? - exactly one character Will find abc1.txt, abck.txt, etc., but not abc.txt
		ls a*.txt	* - 0 to any number of characters Will find a1.txt, ahello.txt, and also a.txt.
		ls a[3-5].txt	Will find a3.txt, a4.txt, and a5.txt
	zip/unzip	zip small bigfile.txt	Zips one file: bigfile.txt is zipped into small.zip
		zip -j -v zipfile mydir/*	Zip all files in the directory mydir into zipfile.zip. -j do not include directory path in zipfile (junk it). -v give information as it proceeds (be verbose).
		zip -r -v ../myzip.zip *	Zip all files and subdirectories of current directory and place resulting zip file one level up (the ../) so it doesn't get involved itself in the zip. -r recurse through all subdirectories -v be verbose
		zip -j -v at032749.zip \ $r/at032[7-9]*.mo \ $r/at033*.mo \ $r/at034*.mo	Zip files at0327*.mo to at0349*.mo junking the path contained in $r.
		unzip myzip	Unzip the zip archive, creating all contained directories, if any, rooted at current directory.

AREA	FUNCTION	COMMAND	ACTION
...files		unzip -aa myzip	While unzipping, convert MS-DOS style carriage return/line feed to just a UNIX style line feed.
		unzip -vlt myzip	Don't unzip. Just list the contents and make sure all contained files are unzippable.
		unzip -j myzip -d mydir	Unzip myzip, but put the files into mydir instead of here, and don't use any contained paths.
Help	commands	man grep	Show help on grep command
		man man	Show help on using man
Permissions	show	ll x.txt	rwx rwx rwx (owner group world) file: r-read; w-write; x-execute directory: r-list files; w-create/delete files; x-cd to dir
	change		Easiest way is to think of each group of rwx as one digit with r=4, w=2, x=1
		chmod 777 x.txt	World accessible and executable: 777=rwx rwx rwx
		chmod 750 myprog	Only I can delete, me and group can execute: 750=rwx r-x ---
		chmod 700 myprog	Only I can execute: 700=rwx --- ---
		chmod 644	Only I can write, anyone can read: 644=rwx r-- r--
		umask 0000	Set default permissions to be used when creating a file or directory to XOR of mask: 0000 gives 777 ie rwx rwx rwx 0133 gives 644 ie rw- r-- r--
Piping		cat myfile | tee x.txt	Display my file on the terminal AND save it to another file at same time.
System	cpu	cat /proc/cpuinfo	Processor cpu info
	date/time	date	Show current date and time
	domain name	hostname -f	Show fully qualified domain name
	IP address	hostname -i	Show system's IP address
	Linux kernel	uname -a	Show system, kernel, processor, etc.
	memory	cat /proc/meminfo	Processor memory info
	uptime	uptime	How long since last reboot
	last reboot	who -a	What time was the last reboot
	run level	who -a	What is the system run level
Users	log out	Ctrl d	Exit system. ("logout" may work on some systems too)
	me	whoami	Show my user name
	password	passwd	Change password
	show users	ps	List my processes
		ps -u userx	Show somebody else's processes
		ps -ef	Show everybody's process fully.
	logged in users	who	Show usernames, tty, address
	my info	who am I	A trick. who with 2 args means who -m, which gives my username, tty, address

Handling special characters

TASK	EXAMPLE	EXPLANATION
Get a newline into a variable	nl=$'\n' echo 'aaa' "$nl" 'bbb' aaa bbb echo ${#nl} 1	There is some setting that causes shells to strip non-printing characters when doing string operations. eg x=$(echo -e "\n") will not put anything into x. Nor will printf to a variable, etc. And if the last character of a file f.txt is a newline, then x=$(tail --bytes=1 f.txt) will not put a newline in x The construct $' ' will preserve special characters. When you use it, surround it with double quotes as shown.
How to get a control character	nl=$'\cj'	This is also a newline.
Escape character	esc=$'\E'	
Any character by code value	xx=$'\xHH'	Using hex digits.
How to enter any character on a keyboard that doesn't have it.	Ctrl/Shift/u Release (see underlined u) 005C Enter Result: \	

Pattern removal

TYPE	USAGE	EXPLANATION
#	${parameter#pattern}	Remove minimum leading pattern
##	${parameter##pattern}	Remove maximum leading pattern
%	${parameter%pattern}	Remove minimum trailing pattern
%%	${parameter%%pattern}	Remove maximum trailing pattern
	x='10%abc' y=${x%\%*} echo $y 10	How to handle special character # or % in string Remove everything after and including '%'

awk - text file processor

TASK	EXAMPLE	EXPLANATION
Find regular text	awk '/123a/' x.txt	Print all lines containing text 123a
Find except	awk '!/123a/' x.txt	Print all lines that DON'T contain text 123a
Find a "meta" character	awk '/\//' x.txt	Print all lines containing a /. Note preceding \ to force / to be interpreted as a real /.
Find this or that	awk '/^D/ || /^H/' x.txt	Print all lines that BEGIN with a D or an H
Find and extract	awk '/^T/ {print substr($0,1,10)}; !/^T/' x.txt > z.txt	Put in file z.txt the first 10 characters of lines that start with T, and all of the remaining lines.
More complex find/extract	awk '/^T0[13]....48711/ {print substr ($0,1,23) "110" substr ($0,27)}; !/^T0[13]....48711' x.txt > z.txt	Find all records that start with T01 or T03 and have 48711 in positions 8-12; change positions 24-26 to 110. Leave other records untouched. (This statement is typed all on one physical line.)
Use begin/end actions Use external program Calculate total field value	awk -f prog.awk x.txt > z.txt *where prog.awk contains:* BEGIN { x=0 } /^T/ { x=x+substr($0,17,2)} END { print "count = " x }	Total the field in positions 17-18 on each record beginning with a T.
Specify record separator	echo $PATH | awk -v RS=':' '{print}'	Change default line separator to ':' so can print your PATH components one on each line
	awk '/tasklist/ {getline; print; getline; print}' x.txt	Find lines containing "tasklist", and print the following two lines.

sed - file filter

TASK	EXAMPLE	EXPLANATION
	`sed -n 's/^[[:blank:]]\+// p' extract.txt`	Condense all leading blank space (spaces and/or tabs) to nothing. Print just the lines affected.
	`sed 's/_// g' lt.txt`	Remove all underscores
	`sed -n ‘10,15 p’ x.txt`	Print lines 10-15 Print just the lines affected.
	`sed -n -s ‘10,15’ *.txt`	Print lines 10-15 from every file.txt The -s means consider as separate files, not one stream (where line numbers would never reset)

vi - text editor

AREA	COMMAND	EXPLANATION
Start	vi x.txt i	Edit new or existing file x.txt. If it's a new file, give the "i" command right away to start inserting text.
Modes	<ESC>	Stop entering text and go to command mode
Move in file	G	Go to end of file
	1G	Go to top of file
Move by windows	^F	Forward one window
	n^F	Forward n windows
	^D	Forward ½ window
	^B	Back one window
	n^B	Back n windows
	^U	Back ½ window
Move in window	H	Home
	M	Middle
	L	Last line
Move in line	0	Beginning of line (zero)
	$	End of line
	+	Beginning of next line
	-	Beginning of previous line
	n|	Column n
	←↑→↓	Arrows should work
Character editing	i	Insert before cursor
	x	Delete current character
	R	Replace current character (and following)
	~	Toggle case
Line editing	O	Open new line before current line (capital letter O)
	o	Open new line after current line
	dd	Delete current line
	J	Join this line and next
	dd ... p	Cut and paste one line
	mz ... d`z ... p	Mark beginning of range Mark end and cut Paste
Search	/	Specify search text
	n	next match
Substitute	:%s/aaa/bbb/g	Replace aaa with bbb everywhere
	:.,$s/aaa/bbb/g	... from current line to last line
	:1,.s/aaa/bbb/g	... from first line to current line
Exit	:q! <Enter>	Quit without saving
	ZZ	Save file and exit

Regular expressions

Summary

c	matches the non-metacharacter c.
\c	matches the literal character c.
.	matches any character including newline.
^	matches the beginning of a string.
$	matches the end of a string.
[abc...]	character list, matches any of the characters abc....
[^abc...]	negated character list, matches any character except abc....
r1|r2	alternation: matches either r1 or r2.
r1r2	concatenation: matches r1, and then r2.
r+	matches one or more r's.
r*	matches zero or more r's.
r?	matches zero or one r's.
(r)	grouping: matches r.
r{n} r{n,} r{n,m}	One or two numbers inside braces denote an interval expression. If there is one number in the braces, the preceding regular expression r is repeated n times. If there are two numbers separated by a comma, r is repeated n to m times. If there is one number followed by a comma, then r is repeated at least n times. Interval expressions are only available if either --posix or --re-interval is specified on the command line.

Above from ‘man awk’. See the following sites for complete details:

http://web.mit.edu/gnu/doc/html/regex_toc.html
http://www.gnu.org/software/gawk/manual/html_node/Regexp.html

Note that regular expressions are normally enclosed within slashes, eg /c/
This has been omitted in the examples.

Escaping special characters

It may be necessary to "escape" (ie, precede with a \) certain characters.
Sometimes it is required because the character has special meaning to the regular expression analyzer.
For example ? means 0 or 1 occurrences of a character. So if you want to look for a literal ?, you have to type \?.

Sometimes it is necessary to escape characters because you are typing the regular expression into a BASH command line, and the character means something special to BASH.
For example { is used to repeat occurrences a specific number of times, but { is also a BASH meta character. So you have to type \{

Usage in a shell script

To use in a shell script, you can use the compound command [[ ]] with the reg expression comparison operator =~ and test the result with $?

```
[[ “12a3” =~ ^[[:digit:]]+$ ]]
echo $?

0 - matches
1 - doesn't match

x=’    <en>hello there</en>’
[[ “$x” =~ ^.*\<en\>.*\</en\> ]]
echo $?
0
```

Regular expression examples

YOU WANT TO MATCH	EXAMPLE EXPRESSION TO USE	WHAT THE EXAMPLE MATCHES
One specific ordinary character	t	The character 't'
One specific meta character	*	The meta character '*' Have to "escape" it with \ to remove special meaning of the meta character
One possible character from a list	[hp8]	h or p or 8 That is, you specify a list of possibilities
One possible character from a pre-defined list	[[:alnum:]]	Same as [a-zA-Z0-9] These are system defined lists, or "classes". The actual class is [:alnum:], and it can only be used inside the list brackets [].
►	[[:alpha:]]	Same as [a-z A-Z]
	[[:blank:]]	A blank (space or tab)
	[[:cntrl:]]	Any control character (0-31 ASCII)
►	[[:digit:]]	0-9
	[[:graph:]]	A printable, visible character (ie blank not included)
►	[[:lower:]]	a-z
	[[:print:]]	Any non control character
	[[:punct:]]	Not a letter, digit, control, or blank
	[[:space:]]	Whitespace: blank, tab,
►	[[:upper:]]	A-Z
	[[:xdigit:]]	Hexadecimal: 0-9, a-f, A-F
	[[:lower:][:digit:]]	a-z or 0-9
One possible character from a range	[g-m]	Any of g, h, i, j, k, l, m
	[3-6]	Any of 3, 4, 5, 6
Any character	.	Any character
Any character except	[^s]	Any character except s
	[^ntz]	Any character except n or t or z
	[^[:digit:]]	Any character except a digit
Several characters	axz	axz
	a[xz]	ax or az
	a.[xz]	aax, abx, acx, adx, ... aaz, abz, acz, adz, ...
	a..c	az9c, a3{c, ...
	[[:lower:]][[:digit:]]	a-z followed by 0-9 (Compare with [[:lower:][:digit:]] above)
	[^(ntz)]	Anything except ntz (Compare with [^ntz] above. The () means treat the enclosed as one regular expression on its own.)
	[^s]kde	kde, but not skde

Repeated expressions	a?	0 or 1 a's (at least! - could be more)
	a*	0 or more a's (any number of a's, including none)
	a+	1 or more a's (at least 1 a)
	a.*9	a, zero or more characters, 9
	[[:alpha:]][[:digit:]]*	an alpha followed by 0 or more digits
	a{25}	25 a's Note that with awk you have to include switch --posix to use {} repetition. And with sed you have to include -r switch.
	[[:digit:]]{4}\-[[:digit:]]{2}\-[[:digit:]]{2}]]	yyyy-mm-dd
Positioning (anchoring)	^abc	abc at the beginning of the line or string
	^..fred	any two characters at the beginning, followed by fred
	abc$	abc at the end of the line or string
	hello..$	hello and then any two characters and then the end
	^$	A blank line
	^[[:digit:]]+$	The whole line is one or more digits
Combining	abc|xyz	abc OR xyz
	abc[^(xyz)]	abc followed by anything except xyz