

UNIX Reference 1

File Commands

COMMAND	DESCRIPTION
ls	directory listing
ls -al	formatted listing with hidden <i>files</i> cd <i>dir</i> - change directory to <i>dir</i>
cd	change to home
pwd	show current <i>directory</i>
mkdir <i>dir</i>	create a directory <i>dir</i>
rm <i>file</i>	delete <i>file</i>
rm -r <i>dir</i>	delete directory <i>dir</i>
rm -f <i>file</i>	force remove <i>file</i>
rm -rf <i>dir</i>	force remove directory <i>dir</i> *
cp <i>file1 file2</i>	copy <i>file1</i> to <i>file2</i>
cp -r <i>dir1 dir2</i>	copy <i>dir1</i> to <i>dir2</i> ; create <i>dir2</i> if it doesn't exist
mv <i>file1 file2</i>	rename or move <i>file1</i> to <i>file2</i> if <i>file2</i> is an existing <i>directory</i> , moves <i>file1</i> into directory <i>file2</i>
ln -s <i>file link</i>	create symbolic <i>link</i> to <i>file</i>
touch <i>file</i>	create or update <i>file</i>
cat > <i>file</i>	places standard input into <i>file</i>
more <i>file</i>	output the contents of <i>file</i>
head <i>file</i>	output the first 10 lines of <i>file</i>
tail <i>file</i>	output the last 10 lines of <i>file</i>
tail -f <i>file</i>	output the contents of <i>file</i> as it grows, starting with the last 10 lines

Process Management

COMMAND	DESCRIPTION
ps	display your currently active <i>processes</i>
top	display all running <i>processes</i>
kill <i>pid</i>	kill processs id pid
killall <i>proc</i>	kill all processes named <i>proc</i> *
bg	lists stopped or background jobs; resume a stopped job in the background

fg	brings the most recent job to foreground
fg n	brings job <i>n</i> to the foreground

File Permissions

COMMAND	DESCRIPTION
chmod <i>octal file</i>	change the permissions of <i>file</i> to <i>octal</i> , which can be found separately for user, group, and world by adding: 4 - read (r) 2 - write (w) 1 - execute (x)
	Examples: chmod 777 - read, write, execute for all chmod 755 - rwx for owner, rx for group and world. For more options, see man chmod .

SSH

COMMAND	DESCRIPTION
ssh <i>user@host</i>	connect to <i>host</i> as <i>user</i>
ssh -p <i>port user@host</i>	connect to <i>host</i> on port <i>port</i> as <i>user</i>
ssh-copy-id <i>user@host</i>	add your key to <i>host</i> for <i>user</i> to enable a keyed or passwordless login

Searching

COMMAND	DESCRIPTION
grep <i>pattern files</i>	search for <i>pattern</i> in <i>files</i>
grep -r <i>pattern dir</i>	search recursively for <i>pattern</i> in <i>dir</i>
<i>command</i> grep <i>pattern</i>	search for <i>pattern</i> in the output of <i>command</i>
locate <i>file</i>	find all instances of <i>file</i>

System Information

COMMAND	DESCRIPTION
date	show the current date and time
cal	show this month's calendar

uptime	show current uptime
w	display who is online
whoami	who you are logged in as
finger <i>user</i>	display information about <i>user</i>
uname -a	show kernel information
cat /proc/cpuinfo	cpu information
cat /proc/meminfo	memory information
man <i>command</i>	show the manual for <i>command</i>
df	show disk usage
du	show directory space usage
free	show memory and swap usage
whereis <i>app</i>	show possible locations of <i>app</i>
which <i>app</i>	show which <i>app</i> will be run by default

Compression

COMMAND	DESCRIPTION
tar cf <i>file.tar files</i>	create a tar named <i>file.tar</i> containing <i>files</i>
tar xf <i>file.tar</i>	extract the <i>files</i> from <i>file.tar</i>
tar czf <i>file.tar.gz files</i>	create a tar with Gzip compression
tar xzf <i>file.tar.gz</i>	extract a tar using Gzip
tar cjf <i>file.tar.bz2</i>	create a tar with Bzip2 compression
tar xjf <i>file.tar.bz2</i>	extract a tar using Bzip2
gzip <i>file</i>	compresses <i>file</i> and renames it to <i>file.gz</i>
gzip -d <i>file.gz</i>	decompresses <i>file.gz</i> back to <i>file</i>

Network

COMMAND	DESCRIPTION
ping <i>host</i>	ping <i>host</i> and output results
whois <i>domain</i>	get whois information for <i>domain</i>

dig <i>domain</i>	get DNS information for <i>domain</i>
dig -x <i>host</i>	reverse lookup <i>host</i>
wget <i>file</i>	download <i>file</i>
wget -c <i>file</i>	continue a stopped download

Installation

COMMAND	DESCRIPTION
./configure make make install	Install from source
dpkg -i <i>pkg.deb</i>	install a package (Debian)
rpm -Uvh <i>pkg.rpm</i>	install a package (RPM)

Shortcuts

COMMAND	DESCRIPTION
Ctrl+C	halts the current command
Ctrl+Z	stops the current command, resume with fg in the foreground or bg in the background
Ctrl+D	log out of current session, similar to exit
Ctrl+W	erases one word in the current line
Ctrl+U	erases the whole line
Ctrl+R	type to bring up a recent command
!!	repeats the last command
exit	log out of current session

UNIX Reference 2

cd (change directory)	
cd myfolder	Changes the current working directory to "myfolder"
cd ..	Go up one level to the current working directory.
cd ../../	Go up two levels to the current working directory.
cd /	Changes the current working directory to the root directory.
cd ~	Changes the current working directory to your home directory.
mkdir (Make directory)	
mkdir Photos	Create a new folder called "Photos" in the current directory.
mkdir /Photos	Create a new folder called "Photos" in the root directory.
mkdir ~/Photos	Create a new folder called "Photos" in your home directory.
ls (list)	
ls	list the file names in the current working directory.
ls -l	list the file names with "long" description/information (size, privileges, etc)
ls -a	list "all" file names in the current working directory including the hidden files.
ls -l *.jpg	list the file names ending in ".jpg" and display it in "long" description format(-l)
cp (copy)	
cp letter.txt newsletter.txt	Copy the file called "letter.txt" and name it "newsletter.txt" in the current directory
cp my.cnf /etc/my.cnf	Copy the file "my.cnf" and put it inside the root -> etc folder.
cp tax05.db ~/Taxes	Copy the file name "tax05.db" and put it inside my home directory -> Taxes folder
cp *.jpg ~/Photos	Copy all the files with ".jpg" extension and put them inside my home directory -> Photos folder
cp -R ~/Docs /backups/Docs Backup'	Copy the entire "Docs" directory from my home page and put it inside the root -> backups and call it "Docs backup" (use quotes if you use folder names with space. example: 'Docs Backup' (-R stands for "Recursive"))

<code>sudo cp -Rp /Users /UsersBackup</code>	Copy the entire "Users" folder including subfolders and files, preserve owner, group, permissions, and timestamps and save the new folder in the root -> UsersBackup location. "-Rp" stands for Recursive Preserve: Recursive: Copy include subfolders and files. Preserve: Preserve owner, group, permissions, and timestamps information. (use "sudo" to get root access temporarily.)
mv (Move or Rename)	
<code>mv /letter.txt ~/letter.txt</code>	Move the file "letter.txt" from the root directory to the home directory.
<code>mv badletter.txt niceletter.txt</code>	Rename the file "badletter.txt" to "niceletter.txt" in the current directory.
<code>mv Pictures Photos</code>	Rename the folder "Pictures" to "Photos" in the current directory.
<code>mv *.jpg ~/Photos</code>	Move all the files with ".jpg" extension and put them inside my home directory -> Photos folder
rm (Remove)	
<code>rm letter.txt</code>	Delete the file "letter.txt" from the current directory.
<code>rm ~/BadPhotos/*.jpg</code>	Delete all the files with the '.jpg' extension inside your home directory -> "BadPhotos" folder.
<code>rm -R Temp</code>	Delete the "Temp" directory and all of its contents in the current directory (-R stands for "Recursive")
<code>rm -fr Temp</code>	Delete the "Temp" directory and all of its contents including write-protected files without prompting in the current directory (-f stands for "force" -r stands for "recursive")
find (files and folders)	
<code>find ~ -name myletter.doc -print</code>	Search for the file names "myletter.doc" inside my home directory and print the result to the screen
<code>sudo find / -name mysql -print</code>	Search for the file and folder names "mysql*" starting from the root directory and everywhere within it and print the result to the screen. (use "sudo" to get root access temporarily.)
<code>find . -name myletter.doc -print</code>	Search for the file names "myletter.doc" inside the current directory and print the result to the screen
<code>find . -name 'myletter*' -print</code>	Search for the file names starting "myletter" inside the current directory and print the result to the screen
locate (similar to find)	
<code>locate ~ -name myletter.doc</code>	Search for the file names "myletter.doc" inside my home directory and print the result to the screen
pwd (print working directory)	
<code>pwd</code>	Displays the pathname of the current working directory.

who (who logged in)	
who	Displays who is logged into the system.
who am i	Displays my user name.
who -uH	Displays who is logged into the system including heading "H" and idle time information.
su (set user) - type exit to switch back to your own identity	
su	Temporarily become the root user. (this will give you root access privileges and the most control over the OS) - it will prompt you for the administrator password.
su username	Temporarily become another user called "username" (replace "username" with the user that you wish to use as your new identity - this will give you access privileges for the "username") - it will prompt you for the that user's password.
sudo (set user and do) - similar to su except 'su' will give you prompt but 'sudo' you can start typing commands right after the 'sudo' command.	
sudo find / -name mysql -print	Temporarily changes your identity to the root user so you can search for all the files including the once that require root access privilege. It prompts you for administrator/root password
sudo Bobuser rm /Users/Bobuser/Photos/myphoto.jpg	Temporarily changes your identity to the "Bobuser" identity so you can delete a photo named "myphoto.jpg" from the home directory -> Photos folder belonging to Bobuser - It prompts you for "Bobuser"'s password.
ps (running processes)	
ps -ax	List all running processes
ps -aux	List detailed information on all running processes.
top (CPU-intensive processes currently running) - press the "q" key to quit the "top" utility	
top	List all running processes sorted by process id - descending and updating every second - don't forget to press the "q" key to quit, otherwise it will run continuously.
top -us10	List all running processes sorted by CPU usage - descending and updating every 10 seconds - don't forget to press the "q" key to quit, otherwise it will run continuously.
kill	
kill 160	Tell the process ID #160 to terminate.
kill -9 160	Terminate the process ID # 160 at once without any hesitation.

UNIX Reference 3

UNIX

Reference

Computing and Information Technology

Basic Commands

Log out of system	<code>logout</code>
Exit current shell	<code>^D</code> or <code>exit</code>

Online Documentation

See online manual page	<code>man command</code>
Search for a manual page	<code>man -k keyword</code>

Files

List filenames	<code>ls</code>
- with hidden files	<code>ls -a</code>
- with file permissions	<code>ls -l</code>
- with group ownership	<code>ls -g</code>
Copy a file	<code>cp old new</code>
Copy a file to dirname	<code>cp file dirname</code>
Rename (move) a file	<code>mv old new</code>
Remove (delete) a file	<code>rm file</code>
Append file1 to file2	<code>cat file1 >> file2</code>
Home directory	<code>~</code>
Home directory of user	<code>~user</code>
Change file permissions	<code>chmod (ugo +-rwx) file</code>
Wild cards	
- single character	<code>?</code>
- multiple characters	<code>*</code>
- range (a and b are single characters)	<code>[a-b]</code>

File Editors

Emacs	<code>emacs file</code>
vi	<code>vi file</code>
pico	<code>pico file</code>

Using less

View file	<code>less file</code>
------------------	------------------------

next line	<Return>
next page	<Space>
search for pattern	/pattern
next occurrence	n
next file	:n
help	:h
quit	:q

Directories

Make a directory	mkdir dirname
Change directories	cd dirname
Remove a directory	rmdir dirname
See the current directory name	pwd
Current directory	.
Parent of the current directory	..
Root of the file system	/

Printing

Print file to default printer (Bell 101)	lpr file
Print file to a printer at another site	lpr -Pprintername file
View printer queue	lpq -Pprinter
Remove job number jn	lprm jn
View job turnaround time	prstat

Job and Process Control

Run job j in the background	j&
List jobs	jobs
Connect to job number n	%n
List all processes for user	/bin/ps -u user
Kill process with id of pid	kill -9 pid
Kill job number n	kill -9 %n

I/O Redirection

Standard output	> or >!
Append to standard output	>> or >>!
Standard input	<
Standard error and output	>&

Standard error separately	<code>< command > output >& errorfile</code>
---------------------------	---

C Shell History

Create history list n items long	<code>set history=n</code>
See history list	<code>history</code>
Repeat last command	<code>!!</code>
Display last command	<code>!!:p</code>
Execute previous command that starts with str	<code>!str</code>
Command line n	<code>!n</code>
First argument of last command	<code>!^</code>
Last argument of last command	<code>!\$</code>
All arguments of last command	<code>!*</code>
Replace old with new in last command	<code>^old^new^</code>

C Shell Customization

Create alias	<code>alias</code>
Set environment variable	<code>setenv</code>
Set shell variable	<code>set</code>

Programming Language Compilers

ada	<code>ada</code>
Allegro Common Lisp	<code>acl</code>
Ansi C	<code>acc</code>
Assembler	<code>as</code>
C++	<code>CC</code>
Fortran	<code>f77</code>
Fortran 90	<code>f90</code>
GNU C	<code>gcc</code>
GNU C++	<code>g++</code>

UNIX Reference 4

UNIX command and scripting reference

Environment control

Command	Description
cd d	Change to directory d
mkdir d	Create new directory d
rmdir d	Remove directory d
mv f1 [BlueGuru:f2...] d	Move file f to directory d
mv d1 d2	Rename directory d1 as d2
pwd	See what directory you are in
passwd	Change password
alias name1 name2	Create command alias
unalias name1	Remove command alias name1
rlogin nd	Login to remote node
logout	End terminal session
setenv name v	Set env var to value v
unsetenv [BlueGuru:name1 name2...]	remove environment variable

Output, communication, and help

Command	Description
lpr -P printer f	Output file f to line printer
script [BlueGuru:f]	Save terminal session to f
exit	Stop saving terminal session
mail username	Send mail to user
biff [BlueGuru:y/n]	Instant notification of mail
man name	UNIX manual entry forname
learn	Online tutorial

--	--

Process control

Command	Description
Ctrl/c *	Interrupt processes
Ctrl/s *	Stop screen scrolling
Ctrl/q *	Resume screen output
sleep n	Sleep for n seconds
jobs Print	list of jobs
kill [BlueGuru:%n]	Kill job n
ps	Print process status stats
ps -ef	List all running processes
kill -9 n	Remove process n
Ctrl/z *	Suspend current process
stop %n	Suspend background job n
command&	Run command in background
bg [BlueGuru:%n]	Resume background job n
fg [BlueGuru:%n]	Resume foreground job n
exit	Exit from shell

Give everyone read and execute permissions to a file

```
chmod 755 filename
```

Give everyone full permissions to a file

```
chmod 777 filename
```

Displays information about total space and available space on a file system

df

Flags

-g Displays statistics in units of GB blocks. The output values for the file system statistics would be in floating point numbers as value of each unit in bytes is significantly high.

-i Displays the number of free and used i-nodes for the file system; this output is the default when the specified file system is mounted.

-l Displays information on the total number of blocks, the used space, the free space, the percentage of used space, and the mount point for the file system.

-k Displays statistics in units of 1024-byte blocks.

-m Displays statistics in units of MB blocks. The output values for the file system statistics would be in floating point numbers as value of each unit in bytes is significantly high.

-M Displays the mount point information for the file system in the second column.

-P Displays information on the file system in POSIX portable format.

-s Gets file system statistics from the VFS specific file system helper instead of the statfs system call. Any arguments given when using the -s flag must be a JFS or Enhanced JFS filesystem mount point or device. The filesystem must also be listed in /etc/filesystems.

-t Includes figures for total allocated space in the output.

-v Displays all information for the specified file system.

Display the number of files in a directory

```
ls -lt | wc -l
```

Display running processes...

for a user

```
ps -ef | grep username
```

for an application

```
ps -ef | grep <application name or part of app name>
```

Kill a running process

```
kill <process-id>
```

to collect the thread dumps use `kill -3 <process-id>`

to kill the process use `kill -9 <process-id>`

Get the active connection in web server user

```
netstat -a|grep -i est*
```

to get the count of the active connection use

```
netstat -a|grep -i est*|wc -l
```

Delete File that have been created before the past 5 Days

During a Support Project you may need to Delete the Logs that have been older than 60 Days the command for that look like

```
find /<replace my_Directory> -mtime +5 -print | xargs rm
```

where mtime is the File last modification time

Determine the last Day of Month

I found this in net.This script is useful in finding the last day of the month.

Sometime it is necessary to run a command or script on the last day of a month.

```
month=`date +%m`  
today=`date +%e`  
year=`date +%Y`  
lastday=`cal $month $year | grep -v "^$" | tail -1 | awk '{print $NF}'`  
if [BlueGuru: $today == $lastday ]; then  
echo your code runs here  
fi
```

The `grep -v` command removes the last line of output if it happens to be blank and the `awk` command prints the last chunk of data on the line.

FTP script tips

The FTP command only give the read write permission to the owner of the file , sometimes it is necessary for the member of the group or the world to read the file or write the file , in this case you will get the File Permission denied error to over come this in your FTP script add this unmask like

This will create the file with rw-rw-rw- permission.
site umask 000

so your FTP scripts will look something like

```
ftp $1 << FTP_CMD
cd <your destination Folder>
site umask 000
put $fileName1
quit
FTP_CMD;
```

UNIX Reference 5

VI

Cursor control and position

h	Left
j	Down
k	Up
l (or spacebar)	Right
w	Forward one word
b	Back one word
e	End of word
(Beginning of current sentence
)	Beginning of next sentence
{	Beginning of current paragraph
}	Beginning of next paragraph
[[Beginning of current section
]]	Beginning of next section
0	Start of current line
\$	End of current line
^	First non-white character of current line
+ or RETURN	First character of next line
-	First character of previous line
n	character <i>n</i> of current line
H	Top line of current screen
M	Middle line of current screen
L	Last line of current screen
nH	<i>n</i> lines after top line of current

Editing

A	Append to end of current line
i	Insert before cursor
I	Insert at beginning of line
o	Open line above cursor
O	Open line below cursor
ESC	End of insert mode
Ctrl-I	Insert a tab
Ctrl-T	Move to next tab position
Backspace	Move back one character
Ctrl-U	Delete current line
Ctrl-V	Quote next character

	screen		
<i>n</i> L	<i>n</i> lines before last line of current screen	Ctrl-W	Move back one word
Ctrl-F	Forward one screen	cw	Change word
Ctrl-B	Back one screen	cc	Change line
Ctrl-D	Down half a screen	C	Change from current position to end of line
Ctrl-U	Up half a screen	dd	Delete current line
Ctrl-E	Display another line at bottom of screen	ndd	Delete <i>n</i> lines
Ctrl-Y	Display another line at top of screen	D	Delete remainder of line
z RETURN	Redraw screen with cursor at top	dw	Delete word
z .	Redraw screen with cursor in middle	d}	Delete rest of paragraph
z -	Redraw screen with cursor at bottom	d^	Delete back to start of line
Ctrl-L	Redraw screen without re-positioning	c/pat	Delete up to first occurrence of pattern
Ctrl-R	Redraw screen without re-positioning	dn	Delete up to next occurrence of pattern
/text	Search for <i>text</i> (forwards)	dfa	Delete up to and including <i>a</i> on current line
/	Repeat forward search	dta	Delete up to, but not including, <i>a</i> on current line
?text	Search for <i>text</i> (backwards)	dL	Delete up to last line on screen
?	Repeat previous search backwards	dG	Delete to end of file
n	Repeat previous search	J	Join two lines
N	Repeat previous search, but in opposite direction	p	Insert buffer after cursor
/text/+n	Go to line <i>n</i> after <i>text</i>	P	Insert buffer before cursor
?text?-n	Go to line <i>n</i> before <i>text</i>	rx	Replace character with <i>x</i>
%	Find match of current parenthesis, brace, or bracket.	Rtext	Replace <i>text</i> beginning at cursor
Ctrl-G	Display line number of cursor	s	Substitute character
<i>n</i> G	Move cursor to line number <i>n</i>	ns	Substitute <i>n</i> characters
: <i>n</i>	Move cursor to line number <i>n</i>	S	Substitute entire line
G	Move to last line in file	u	Undo last change
		U	Restore current line
		x	Delete current cursor position
		X	Delete back one character
		<i>n</i> X	Delete previous <i>n</i> characters
		.	Repeat last change
		~	Reverse case
		y	Copy current line to new buffer
		yy	Copy current line
		"xyy	Copy current line into buffer <i>x</i>
		"Xd	Delete and append into buffer <i>x</i>

"xp	Put contents of buffer x
y]]	Copy up to next section heading
ye	Copy to end of word

File Handling

:w	Write file
:w!	Write file (ignoring warnings)
:w! <i>file</i>	Overwrite <i>file</i> (ignoring warnings)
:wq	Write file and quit
:q	Quit
:q!	Quit (even if changes not saved)
:w <i>file</i>	Write file as <i>file</i> , leaving original untouched
ZZ	Quit, only writing file if changed
:x	Quit, only writing file if changed
:n1,n2w <i>file</i>	Write lines <i>n1</i> to <i>n2</i> to <i>file</i>
:n1,n2w >> <i>file</i>	Append lines <i>n1</i> to <i>n2</i> to <i>file</i>
:e <i>file2</i>	Edit <i>file2</i> (current file becomes alternate file)
:e!	Reload file from disk (revert to previous saved version)
:e#	Edit alternate file
%	Display current filename
#	Display alternate filename
:n	Edit next file
:n!	Edit next file (ignoring warnings)
:n <i>files</i>	Specify new list of <i>files</i>
:r <i>file</i>	Insert <i>file</i> after cursor
:r ! <i>command</i>	Run <i>command</i> , and insert output after current line

UNIX Reference 6

VI

Unix vi Reference Card

	arrow keys move cursor	^n, ^p, <SP>, <BS> same as arrows
<ESC>	escape from text entry	^g tell what is going on
:	give ex command	ZZ write and quit
<CR>	start of next line	- start of previous line
^b	go back one page	^u scroll back 1/2 page
^f	go forward one page	^d scroll forward 1/2
page		or end indent

<code>^e</code>	scroll down one line	<code>^y</code>	scroll up one line
<code>H</code>	home screen line	<code>L</code>	last screen line
<code>x</code>	delete character	<code>r</code>	replace char with next typed
<code>a</code>	append after cursor	<code>A</code>	append after line
<code>i</code>	insert before cursor	<code>I</code>	insert before line
<code>p</code>	put before cursor	<code>P</code>	put before line
<code>u</code>	undo changes	<code>U</code>	undo changes in this line only
<code>c obj</code>	change object to following text		

Object defining characters

<code>w</code>	forward one word	<code>W</code>	forward, ignore punctuation
<code>b</code>	backward one word	<code>B</code>	backward, ignore punc.
<code>e</code>	end of current word		<CR>,- as above

Characters using buffers

<code>d obj</code>	delete object	<code>dd</code>	delete line
<code>y obj</code>	yanks object	<code>yy</code>	yank line
<code>p</code>	place what is in buffer		
<code>"</code>	select buffer		

NOTE: There are 26 buffers, each corresponding to a letter of the alphabet. To refer to a specific buffer type "<letter> e.g. "a refers to buffer a. Therefore "add would delete the line that cursor is on and place it into buffer a. To place the deleted line elsewhere, position cursor and type "ap.

In order to delete or yank a section of text, you must place a mark to where the deletion/yank is to stop. This is done by typing: m<letter> (do not use the same buffer to place a mark and store text). To return to that location, type `<letter>. When deleting or yanking to a mark, say b, position the cursor at the beginning of the desired text, and type "ad`b. This deletes the text from cursor until mark b, and places it in buffer a. "ad'b will delete until the end of the line in which mark b is (note difference in ` and ').

Search

<code>/text<CR></code>	search forward for text	<code>?text<CR></code>	search backward
<code>n</code>	next occurrence	<code>N</code>	next occurrence in opposite direction

Miscellaneous

<code>J</code>	adjoin current line to line above
<code>!!</code>	place results of command into file i.e. !!cat it.p will place the list of it.p in your file.

Unix ex Reference Card

```
n      set pointer to line n
n,n    specify a range of lines for next command
$      used to indicate last line in file
.      print current line or end insertion

p      print specified lines
p#     print specified lines with line numbers
a      append lines after this line; end insertion with "."
i      insert lines after this line; end insertion with "."
d      delete specified lines
u      undo last deletion or substitution
c      change (i.e. replace) specified lines to new text
ya     yank specified lines into buffer
pu     lines from buffer before this one
w      write current text to file
q      quit edit
q!     forces quit without write
wq     write and quit
vi     enter vi editor
se     display settings; refer to manual for details

/text/      search for text
//          search for next occurrence
s/oldtext/newtext/  substitute newtext for oldtext in
                  specified lines
n1,n2copy3  copy lines n1-n2 to after n3
global/pattern/cmds  search file for pattern and do cmds on those
                  lines.

set nu      insert line numbers
set nonu    remove line numbers
```

--

"...nothing kills that does not know ye."

-Meg Davis, 'The Elf Glade'

UUCP: ihnp4!gargoyle!sphinx!d757 d757@sphinx.uchicago.edu
Lawrence Lerner University of Chicago Computation Center

UNIX Reference 7

Regular Expression Metacharacters

.	Any one character
[...]	Any one of the characters within the square brackets
[^...]	Any one of the characters <i>not</i> within the square brackets
^	Start of line
\$	End of line
\<	Start of word
/>	End of word
(vertical bar)	Separates two expressions, matches either
?	Previous character (or group) is optional
+	One or more of the previous character (or group)
*	Any number (including none) of the previous character (or group) NOTE: Matches as many as possible Three uses:
()	1: Used to enclose a pair of expressions, separated by (vertical bar - see above) 2: Grouping for quantifiers ('?', '+', and '*' - see above) 3: Carry some text that matches the expression within (see '\1', etc, below)
\1 (and \2, \3, etc)	Output the text 'carried forward' by the brackets (see '()' above).

UNIX Reference 8

AWK built-in function reference

break	Leave the current 'while' or 'for' loop.
close(<i>filename expression or command expression</i>)	Close the file which was opened by the specified expression.
continue	Start the next iteration of the current 'while' or 'for' loop.
delete(<i>array[element]</i>)	Delete the specified <i>element</i> of the <i>array</i> .
do <i>statements</i> while (<i>expressions</i>)	Execute <i>statements</i> , then evaluate <i>expression</i> . Repeat until <i>expression</i> evaluates false.
exit	Execute the END instructions, then exit the program.
for (<i>expr1</i> ; <i>expr2</i> ; <i>expr3</i>) <i>statements</i>	<i>expr1</i> is evaluated - this is usually use to set up the conditions for the loop. If <i>expr2</i> evaluates true, the <i>statements</i> are run. <i>expr3</i> is then run - this is often used to increment a counter. The construct then loops, checking <i>expr2</i> again. When <i>expr2</i> evaluates false, the loop terminates.
function <i>fname(parameters)</i> {	Define a function.

<code>statements</code> <code>}</code>	
<code>getline [varname]</code> <code>[<filename]</code>	Get a line from the specified file. If <i>varname</i> is not specified, the line is loaded into \$0. If <i>varname</i> is specified, the variable is populated instead of \$0. See also additional syntax below.
<code>statements getline</code> <code>[varname]</code>	The <i>statements</i> are run, and the output is passed to <code>getline</code> . <i>varname</i> operates as described above.
<code>gsub (regex,</code> <code>newstring, string)</code>	Replace every match of the regular expression <i>regex</i> in <i>string</i> with <i>newstring</i> .
<code>if (condition)</code> <code>statements</code> <code>[else</code> <code>statements]</code>	If <i>condition</i> evaluates true, the first block of <i>statements</i> are run. If not, the <i>statements</i> in the optional 'else' section are run.
<code>index(substring,</code> <code>string)</code>	Returns the index of <i>substring</i> within <i>string</i> , or 0 if not present.
<code>length([string])</code>	Returns the length of <i>string</i> , or the length of \$0 if <i>string</i> not specified.
<code>match(string, regex)</code>	Returns the position of the first match for the regular expression <i>regex</i> in <i>string</i> , or 0 if no matches are found. Sets RSTART and RLENGTH variables.
<code>next</code>	Ignore any further instructions for this record. Read the next record, and process it.
<code>print [expressions]</code> <code>[>destination]</code>	Print the <i>expressions</i> (or the current line if no expression given) to the specified <i>destination</i> (the standard output, if not specified).
<code>printf format</code> <code>expressions</code> <code>[>destination]</code>	Printf the <i>expressions</i> formatted according to <i>format</i> to the specified <i>destination</i> , or standard output if none specified.
<code>return [expression]</code>	Return from a function, returning the value of <i>expression</i> to the caller.
<code>split(string, array</code> <code>[,regex]</code>	Split the <i>string</i> into components, populating the <i>array</i> with the results. Use the regular expression <i>regex</i> to specify the split boundary, or the contents of the variable FS is <i>regex</i> if not specified.
<code>sprintf format</code> <code>expressions</code>	As <code>printf</code> , except that the formatted string is returned instead of printed.
<code>sub (regex,</code> <code>newstring, string)</code>	As <code>sub</code> , except only the first match is changed.
<code>substr(string, start</code> <code>[,length]</code>	Return <i>length</i> characters from the specified <i>string</i> , starting from <i>start</i> . If <i>length</i> is not specified, return rest of record.
<code>system(command)</code>	Execute the <i>command</i> , and return the exit status.
<code>tolower(string)</code>	Return the string with all upper case characters replaced with their lower case equivalents.
<code>toupper(string)</code>	Return the string with all lower case characters replaced with their upper case equivalents.
<code>while(condition)</code> <code>statements</code>	While the specified <i>condition</i> is true, execute the <i>statements</i> . Re-evaluate the <i>condition</i> before each execution of the <i>statements</i> .

AWK built-in variable reference

ARGC	Number of command-line arguments.
ARGIND	Index (within ARGV) of file currently being processed.

ARGV	Array of command-line arguments.
CONVFMT	Conversion format for numbers in 'printf' syntax.
ENVIRON	Array containing values of current environment.
ERRNO	String containing text version of most recent error.
FIELDWIDTHS	List of field widths separated by white spaces. If specified, is used by GNU awk instead of FS to parse records into fields.
FILENAME	Name of input file.
FNR	Record number in input file.
FS	Field separator.
IGNORECASE	If non-zero, many functions will ignore case when doing comparisons.
NF	Number of fields in current record.
NR	Number of records processed.
OFMT	Output format for numbers in 'printf' syntax.
OFS	Output field separator.
ORS	Output record separator.
RS	Input record separator.
RT	Record terminator.
RSTART	Index of first character matched by a successful call to the match() function.
RLENGTH	Length of string matched by a successful call to the match() function.
SUBSET	Character used to separate multiple items in arrays.

UNIX Reference 9

AWK Reference

Examples

- `awk '{print $1, $3}' file`
Prints the first and third columns of *file*.
- `ls -lg dir | awk '{ x += $5 } END { print "total K-bytes: " (x + 1023)/1024 }'`
Prints the total number of kilobytes used by *dir*.
- `awk 'NR > 20' file`
Prints *file* starting at line 21.
- `awk 'NF > 0' file`
Prints all lines in *file* which have at least one field, e.g. blank lines are ignored.
- `awk 'length($0) > 80' file`
Prints all lines in *file* longer than 80 characters.
- `awk 'END { print NR }' file`
Counts the number of lines in *file*.
- `awk 'BEGIN{ for(item in ENVIRON) print item, ENVIRON[item]}'`
Print the environment variables (gawk only).

Variables

ARGC	Number of arguments passed on the command line.	(nawk)
ARGIND	Index in `ARGV' of the current file.	(gawk)
ARGV	An array containing command-line arguments.	(nawk)
CONVFMT	C-style format to convert numbers to strings.	
ENVIRON	Associative array with environment variables.	(gawk)
ERRNO	Error description string	(gawk)
FILENAME	The current filename.	
FNR	The record number in the current file.	(nawk)
FS	The field separator.	
IGNORECASE	Boolean indicating case-independent matching.	(gawk)
NF	Number of fields in the current record.	
NR	Number of the current record (e.g. line number).	
OFMT	Number conversion format when using print.	(nawk)
OFS	Output field separator.	
ORS	Output record separator.	
RLENGTH	Length of string matched.	(nawk)
RS	Record separator.	
RSTART	Position in string of match.	(nawk)
RT	Input text matching text denoted by `RS'.	(gawk)
SUBSEP	Separator character for array subscripts.	(nawk)
\$0		
\$n		

UNIX Reference 10

AN AWK PRIMER/AWK QUICK REFERENCE GUIDE

From Wikibooks, the open-content textbooks collection

< [An Awk Primer](#)

Jump to: [navigation](#), [search](#)

[An Awk Primer](#)

This final section provides a convenient lookup reference for Awk programming.

- Invoking Awk:

```
awk [-F<ch>] {pgm} | {-f <pgm file>} [<vars>] [-|<data file>]
```

-- where:

```
ch:          Field-separator character.      pgm:          Awk
command-line program.      pgm file:      File containing an Awk program.
vars:         Awk variable initializations.  data file:    Input data
file.
```

- General form of Awk program:

```
BEGIN          {<initializations>}          <search pattern 1>
{<program actions>}      <search pattern 2> {<program actions>}
...      END          {<final actions>}
```

- Search patterns:

```
/
```

The search can be constrained to particular fields:

```
$<field> ~ /<string>/      Search for string in specified field.
$<field> !~ /<string>/      Search for string \Inot|i in specified field.
```

Strings can be ORed in a search:

```
/(<string1>)|(<string2>)/
```

The search can be for an entire range of lines, bounded by two strings:

```
/<string1>/,/<string2>/
```

The search can be for any condition, such as line number, and can use the following comparison operators:

```
== != < > <= >=
```

Different conditions can be ORed with "||" or ANDed with "&&".

```
[<charlist or range>]      Match on any character in list or range.
[!<charlist or range>]      Match on any character not in list or range.
.                          Match any single character.      *
Match 0 or more occurrences of preceding string.      ?
Match 0 or 1 occurrences of preceding string.      +
Match 1 or more occurrences of preceding string.
```

If a metacharacter is part of the search string, it can be "escaped" by preceding it with a "\".

- **Special characters:**

```
\n      Newline (line feed).
      Backspace.  \r      Carriage return.  \f      Form feed. A
"\\" can be embedded in a string by entering it twice: "\\\".
```

- **Built-in variables:**

```
$0; $1,$2,$3,...  Field variables.      NR      Number of
records (lines).  NF      Number of fields.  FILENAME
Current input filename.  FS      Field separator
character (default: " ").  RS      Record separator
character (default: "\n").  OFS     Output field
separator (default: " ").  ORS     Output record
separator (default: "\n").  OFMT    Output format
(default: "%.6g").
```

- **Arithmetic operations:**

```
+  Addition.      -  Subtraction.      *  Multiplication.      /
Division.        %  Mod.      ++  Increment.      --  Decrement.
```

Shorthand assignments:

```
x += 2  -- is the same as: x = x + 2    x -= 2  -- is the same
as: x = x - 2    x *= 2  -- is the same as: x = x * 2    x /= 2  --
is the same as: x = x / 2    x %= 2  -- is the same as: x = x % 2
```

- **The only unique string operation is concatenation, which is performed simply by listing two strings connected by a blank space.**

- **Arithmetic functions:**

```
sqrt()      Square root.      log()      Base \Ie\i log.      exp()
Power of \Ie\i.  int()      Integer part of argument.
```

- **String functions:**

- **length()**

```
Length of string.
```

- `substr(<string>,<start of substring>,<max length of substring>)`

Get substring.

- `split(<string>,<array>,[<field separator>])`

Split string into array, with initial array index being 1.

- `index(<target string>,<search string>)`

Find index of search string in target string.

- `sprintf()`

Perform formatted print into string.

- Control structures:

```
if (<condition>) <action 1> [else <action 2>]   while
(<condition>) <action>
```

```
for (<initial action>;<condition>;<end-of-loop action>) <action>
```

Scanning through an associative array with "for":

```
for (<variable> in <array>) <action>
```

Unconditional control statements:

```
break      Break out of "while" or "for" loop.   continue
Perform next iteration of "while" or "for" loop.  next      Get
and scan next line of input.   exit        Finish reading input and
perform END statements.
```

- Print:

```
print <i1>, <i2>, ...   Print items separated by OFS; end with
newline.   print <i1> <i2> ...   Print items concatenated; end
with newline.
```

- `Printf()`:

General format:

```
printf(<string with format codes>,[<parameters>])
```

Newlines must be explicitly specified with a `"\n"`.

General form of format code:

```
%[<number>]<format code>
```

The optional "number" can consist of:

- A leading `"-"` for left-justified output.
- An integer part that specifies the minimum output width. (A leading `"0"`

causes the output to be padded with zeroes.)

- A fractional part that specifies either the maximum number of characters to be printed (for a string), or the number of digits to be printed to the right of the decimal point (for floating-point formats).

The format codes are:

```
d    Prints a number in decimal format.    o    Prints a number in
octal format.    x    Prints a number in hexadecimal format.    c
Prints a character, given its numeric code.    s    Prints a string.
e    Prints a number in exponential format.    f    Prints a number
in floating-point format.    g    Prints a number in exponential or
floating-point format.
```

- Awk can perform output redirection (using `>` and `>>`) and piping (using `|`) from both `"print"` and `"printf"`.

Retrieved from

["http://en.wikibooks.org/wiki/An_Awk_Primer/Awk_Quick_Reference_Guide"](http://en.wikibooks.org/wiki/An_Awk_Primer/Awk_Quick_Reference_Guide)