UNIX COMMAND REFERENCE LIST

Sands Consulting Pty. Ltd.

Shell Variables

Items in small caps are korn shell built-in variables. Items marked with an * are set by the developer login process.

Variable	Usage
#	Number of command line arguments
-	Options given to shell on invocation
?	Exit status
$	Process Id
_	Last argument to previous command
!	Process Id of last background command
BOLD*	Turn on Bold Mode
DIRSTACK*	pushd/popd functions
EDITOR*	Sets the editor
ENV*	Run this file every time a new korn shell is started
FCEDIT	Default editor for fc command
FEDPATH*	Sets the path that fed searches for files
GPOF*	Turn on Graphics Mode
GPON*	Turn of Graphics Mode
HISTFILE*	Stores the command history
HISTSIZE*	history buffer size
HOME	Home (login) directory
LC_xxx	Environment variables that affect sorting within unix
LINENO	Number of line in script or function just executed
LPDEST*	Default printer
MAIL+	Name of file to check for new mail
OLDPWD	Previous working directory
OPTARG	Argument to option being processed by getopts
OPTIND	Number of first argument after options
ORACLE_BASE	Root directory for all Oracle versions
ORACLE_HOME	Root directory for Oracle command files for the current version
ORACLE_LPARGS*	Printer arguments to use when printing from Oracle
ORACLE_LPPROG*	Printer program to use when printing from Oracle
ORACLE_PATH*	Oracle search path
ORACLE_SID*	Oracle Instance
ORACLE_TERM*	Terminal emulation used by Oracle Applications
PAGER*	Sets the default pager
PATH	Command search path
PPID	Parent Process Id
PRINTER*	Default printer
PS0*	Base prompt that holds unchanging information
PS1*	The actual prompt
PS2*	The continuation prompt
PS3*	select prompt
PS4*	xtrace prompt
PUBLIC	Location of toolset
PWD	Current working directory
RANDOM	Random number between 0 and 32767
REPLY	Users response from select command; default variable for read command
RMSO*	Turn off Reverse Mode
RMUL*	Turn off Underline Mode
SECONDS	Age of shell
SGR0	Turn off all Attributes
SHARE	(suspect this is used by the samba process and points at the shared area)
SHELL	Full pathname of shell
SHELL_DEPTH*	A numeric value that indicates the depth of the current shell.
SMSO*	Turn on Reverse Mode

Variable	Usage
SMUL*	Turn on Underline Mode
TMOUT+	If set, the number of seconds between commands after which the shell automatically terminates.
USER*	current unix user
VISUAL*	command mode editor

Substitution Operators

If the colon ':' is omitted then the test changes from "exists and is not null" to "exists". In other words the operator only tests for existence. A variable with a null value will return null.

Operator	Meaning
${*var*:-*word*}	If *var* exists and isn't null, return its value; otherwise return *word*. ${count:-0} evaluates to 0 if count is undefined.
${*var*:=*word*}	If *var* exists and isn't null, return its value; otherwise set it to *word* and then return its value.* ${count:=0} sets count to0 if it undefined.
${*var*:?*msg*}	If *var* exists and isn't null, return it's value; otherwise print *var*: followed by *msg*, and abort the current command or script. Omitting *msg* produces the default message "parameter null or not set". ${count:? "undefined! "} prints "count: undefined!"
${*var*:+word}	If *var* exists and isn't null, return *word*; otherwise return null. ${count:+1} returns 1 (which could mean "true") if count is defined. This is also useful when an option has to be specified when invoking another command based on the value of a variable. cmd ${file:+ -f $file}

Pattern-matching Operators

The examples assume that var=my.long.file.name

Option	Meaning
${*var#pattern*}	If the pattern matches the beginning of the *var*, delete the shortest part that matches and return the rest. ${var#*.} returns 'long.file.name'
${var##*pattern*}	If the pattern matches the beginning of the *var*, delete the longest part that matches and return the rest. ${var##*.} returns 'name'
${var%*pattern*}	If the pattern matches the end of the *var*, delete the shortest part that matches and return the rest. ${var%.*} returns 'my.long.file'
${*var%%pattern*}	If the pattern matches the end of the *var*, delete the longest part that matches and return the rest. ${var%%.*} returns 'my'

Shell Regular Expression Operators

Option	**Meaning**
* (*exp*)	0 or more occurrences of *exp*
+(*exp*)	1 or more occurrences of *exp*
?(*exp*)	0 or 1 occurrences of *exp*
@(*exp1* | *exp2* | …)	*exp*1 or *exp2* or …
!(*exp*)	Anything that doesn't match *exp*

Shell options

These options control the behaviour of the shell itself. To turn options on use:

set -o *optionList*

To turn options off use:

set +o *optionList*

optionList is one or more options separated by a space.
The abbreviation is to provide Bourne shell compatibility.
The options with an asterisk* are on by default.

Option	**Abbrev**	**Meaning**
allexport	-a	Export all subsequently defined variables
errexit	-e	Exit the shell when a command exits with non-0 status.
bgnice*		Run all background jobs at decreased priority
emacs		Use emacs-style command-line editing
gmacs		Use *emacs*-style command-line editing, but with a slightly different meaning for CTRL-T
ignoreeof		Disallow CTRL-D to exit the shell
markdirs		Add / to all directory names generate from wildcard expansion
monitor*	-m	Enable job control
noclobber		Don't allow > redirection to existing files
noexec	-n	Read commands and check for syntax errors, but don't execute them
noglob	-f	Disable wildcard expansion
nolog		Disable command history
nounset	-u	Treat undefined variables as errors, not as null
privileged	-p	Script is running in *suid* mode
trackall	-h	Substitute full pathnames for commands alias expansions
verbose	-v	Print commands (verbatim) before running them
vi		Use *vi*-style command-line editing
viraw		Use *vi* mode and have each keystroke take effect immediately
xtrace	-x	Print commands (after expansions) before running them

Redirection operators

allow manipulation of various file descriptors
fd0 is stdin
fd1 is stdout
fd2 is stderr

Option	**Meaning**
> *file*	redirect stdout to *file*
< *file*	redirect *file* to stdin
>> *file*	append to *file*
<< *file*	label (hereis document)
>| *file*	over-ride noclobber shell option
<> *file*	use *file* for both stdin and stdout
m>&*n*	route file descriptor *m* to the same place as file descriptor *n*
m<&*n*	duplicate file descriptor *m* from file descriptor *n*
<&-	close stdin
>&-	close stdout
|&	start background process as a coroutine
>&p	direct background process stdout to parents’ stdout
<&p	direct parents’ stdin to background process stdin
|	pipe stdout of process 1 to stdin of process 2

((math operators))

Integer tests are performed between double parentheses. Example:

```
if (( 2 * 6 > 11 )) ; then
```

Arithmetic expressions are indicated with the $((expression))

Quoting is implicit in both cases.

Assignment can be done by either:

```
(( intvar=expression ))
```

or

```
let intvar='expression'
```

Arithmetic Operators	**Meaning**
+	Addtion
-	Subtraction
*	Multiplication
/	Division (decimal part discarded)
%	Modulus
<<	Shift left
>>	Shift right
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT
^	Bitwise XOR

Relational Operators	**True if …**
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equals
!=	Not equals
&&	Logical AND
||	Logical OR

[[operators]]

These operators are used within the [[…]] construct. They can be logically combined with && (“and”) and || (“or”) and grouped with parenthesis. The parenthesis may need to be escaped.
The ! may be used to negate the result.
See also **test**

Operator	**True if…**
-a *file*	*file* exists
-b *file*	*file* exists and is a block special *file*
-c *file*	*file* exists and is a character special *file*
-d *file*	*file* exists and is a directory
-f *file*	*file* exists and is a regular *file*
-g *file*	*file* exists and has set-GID bit set
-k *file*	*file* exists and has sticky bit set
-n string	*string* is of non-zero length
-o *option*	*option* is set
-p *file*	*file* exists and is a named pipe
-r *file*	*file* exists and is readable
-s *file*	*file* exists and has non-zero size
-t *N*	File descriptor *N* points to a terminal
-u *file*	*file* exists and has set-UID bit set
-w *file*	*file* exists and is writable
-x *file*	*file* exists and is executable
-z *string*	*string* has zero length
-G *file*	*file’s* group ID is the same as that of the shell
-L *file*	*file* is symbolic link
-O *file*	*file* is owned by the shell’s user id.
-S *file*	*file* is a socket
*file*A-nt *file*B	fileA is newer than file B
*file*A-ot *file*B	*file*A is older than *file*B
*file*A-ef *file*B	*file*A and *file*B point to the same file
n1-eq *n2*	integer *n1* and *n2* are equal.
n1-ge *n2*	integer *n1* ≥ and *n2*
n1 –gt *n2*	integer *n1* > *n2*
n1 –le *n2*	integer n1 ≤ *n2*
n1-lt *n2*	integer *n1* < *n2*
n1-ne *n2*	integer *n1* and *n2* are not equal.
s = *pattern*	*s* matches *pattern* (which can contain wildcards)
s != *pattern*	*s* does not match *pattern* (which can contain wildcards)
s1 < *s2*	*s1* comes before *s2* in dictionary order.
s1 > *s2*	*s1* comes after *s2* in dictionary order

awk	**Process file according to patterns**	
	awk [options] [prog] [params] [files	stdin if no files specified
	See awk and sed specific area	

Option	**meaning**
-f p*file*	use p*file* as the source of the program
-F*c*	field separator character is *c*
prog	program line (should be in single quotes)
params	form x=…, y=…

cat	**Concatenate and print files**	
	cat [options] [files]	stdin read if – or no files specified

Option	**Meaning**
-e	w/v shows $ before each newline
-s	silent about nonexisting files
-t	w/v shows tabs as ^ l
-u	unbuffered
-v	show non-printing characters (except tabs, newlines and formfeeds)

cd	**Change Directory**	
	cd [directory]	Environment variable $HOME used if no directory

chgrp	**Change Group ID of files (See chown)**
	chgrp group files

Option	**meaning**
group	group name or number

chown	**Change Owner of files**

Option	**meaning**
owner	login name or decimal user-id

cmp	**Compare Two Files**	
	cmp [options] file 1 file 2	stdn read if – specified for file 1

Option	**meaning**
-1	print byte number and bytes
-s	silent, return codes only

cp	**Copy Files**	
	cp file 1file2	make a copy of file 1 named file2
	cp files directory	make copies of specified files in directory

cut	**Cut Out Fields of File**	
	cut [-clist] [files]	
	cut [-flist] [-dchar] [-s] [files]	stdn read if no files specified

Option	**meaning**
-c*list*	pass specified character positions
-d*c*	specify field delimiter (tab default)
-f*list*	pass specified fields
-s	suppress lines with no delimiters
list	comma-seperated list with optional – to indicate a range

date	**Print Current Date**	
	date [+format]	output can be formulated with special characters

Option	**meaning**
%a	abbreviated weekday –Sun to Sat
%d	day- 01 to 31
%D	date as mm/dd/yy
%h	abbreviated month – Jan to Dec
%H	hour – 00 to 23
%j	julian date – 001 to 366
%m	month – 01 to 12
%M	minute – 00 to 59
%n	insert newline
%r	time in AM/PM notation
%s	second – 00 to 59
%t	insert tab character
%T	time as hh:mm:ss
%w	day of week – Sun = 0
%y	last 2 digits of year – 00 to 99

du	**Summarize Disk Usage**
	du [options] [directories]

Option	**meaning**
-a	generate entry for each file (directories only default)
-r	complain about directories that can’t be read
-s	only display a grand total summary

echo	**Echo Arguments**	
	echo [*args*]	Note: special escape conventions (quote *args*)

Sequence	**Character printed**
\b	backspace
\c	omit final newline
\f	form feed
\n	newline
\0*num*	octal value (8 bits), *num* must start with 0
\r	carriage return
\t	tab
\\\\	backslash

expr	**Evaluate Expression Arguments**
	expr args

Please see the man pages for this very useful routine

find	**Find Files**	
	find pathname-list expression	Expressions formed from one or more primaries

Option	**meaning**
-atime *n*	true if file found was accessed in *n* days
-cpio *dev*	the file is written on *dev* in cpio format
-ctime *n*	true if file found was changed in *n* days
-depth	always true; causes entries in directory to be acted on before the directory itself
-exec *cmd*	execute cmd, true if successful exit status
-group *name*	true if file found is owned buy the group *name*
-links *n*	true if file found has n links
-local	true if file is on local system
-mount	don't cross mounted file systems, always true
-mtime *n*	true if file found was modified in n days
-name *file*	true if file matches name of file found
-newer *file*	true if file found modified later than *file*
-ok *cmd*	like-exec except user prompted first
-perm *octal*	true if permission of file is *octal*
-print	print name of files found, always true
-size *n*	true if file found is *n* blocks long
-type *c*	true if file found is: b block special file c character special file d directory f fifo or named pipe p plain file
-user *name*	true if file found is owned by the user *name*
\(*expr* \)	true if *expr* is true, used for grouping
n	*n* means exactly *n*, +*n* means no more than *n*, -*n* means less than *n*

Ways to join primaries:

! *expr*	negate truth value of *expr*
exp1 exp2	true if both *exp1* and *exp2* are true
exp1-o *exp2*	true if either *exp1* or *exp2* is true

getopts	**Parse Command Options**	
	getopts *string* var [*args*]	*args* default to command line

Option	**meaning**
args	parse *args*
var	shell variable to place next option in
string	list of recognised option letters; followed by : if option has argument. begin list with : to suppress error messages.

grep Search File for Pattern

grep [*options*] pattern [*files*] stdin read if no files specified

Option	meaning
-b	precede in line with block number
-c	print count of matching lines only
-i	ignore case of letters in comparisons
-l	print only names of files with matching lines
-n	print line numbers
-s	suppress file error messages
-v	print non-matching lines

kill Terminates or send a Signal to Process

kill [*SigNr*] *pids*

It is strongly recommended that you do not use the numeric values.
The exit status of a killed process is typically 128 + the value of the kill signal.
pids are the process ids to receive the signal. A pid of 0 implies that all processes resulting from current login will be killed.

SigNr	Abbrev	Meaning
01	HUP	hang up
02	INT	interrupt
03	QUIT	quit
04		illegal instruction
05		trace trap
06		IOT instruction
07		EMT instruction
08		floating point exception
09	KILL	kill (cannot be caught or ignored0
10		bus error
11		segmentation violation
12		bad system call argument
13		write on unread pipe
14		alarm clock
15	TERM	software termination signal
16		user defined signal 1
17		user defined signal 2

ln	**Make Links to Files (see CP)**	
	ln [option] *file1 file2*	make a link of *file1* named *file2*
	ln [option*]* *files* directory	make links of specified files in directory

Option	**meaning**
-f	force link despite target file permissions
-s	create symbolic link

lp	**Send Request to Line Printer**	
	lp [*options*] [*files*]	stdin read if '–' or no *files* specified

Option	**meaning**
-c	copy rather than link files
-dptr	sent request to specified printer
-m	send mail after printing complete
-n*n*	print *n* copies (1 default)
-o*option*	specify printer or class dependent option
-s	suppress messages from lp
-t*title*	print title on banner page of printout
-w	write to user's terminal after printing is complete

ls	**List contents of directories**	
	ls [options] [directories]	Defaults to working directory

Option	**Meaning**
-a	List all entries including those starting with .
-A	As for –a but the **.** and **..** entries are not included
-b	Print non-printing characters in octal
-c	Use time file was created when processing -t and -l options
-C	Multi column output with entries sorted down the column
-d	List only the name of the directory, not its' contents
-f	Interpret each argument as a directory
-F	Add **/** to directories and ***** to executable files
-g	Like -l but don't print the owner
-i	Print the i-node number
-l	Long list. (mode, links, owner, group, size time of last modification)
-m	Print files across the line separated by commas (**,**)
-n	Like -l but use numeric user and group ids
-o	Like -l but don't print the group
-p	Add / to directories
-q	Print non-printing characters as a **?**
-r	Reverse sort order
-R	Recursively list sub-directories
-s	Print size of file in blocks
-t	Sort by modification time
-u	Use time of last access when processing -t and -l options
-x	Multi-column list sorted across each row

man	**Print Manual Entries**
	man [*options*] [*sections*] *titles*

Option	**meaning**
-12	produce 12 pitch output
-c	invoke col to process output
-d	search current directory instead of /usr/manV
-s	typeset in small (6”x 9”) format
-t	typeset in default (8.5” x 11”) format
-T*term*	format using nroff for terminal type *term* *(450 default)*
-w	print only pathnames of entries
-y	use non-compacted macros

mv	**Move Files(See CP)**	
	mv [*option*] *file1 file2*	rename (or move) *file* 1 to *file2*
	mv [option] *files* directory	rename (or move) specified files to directory

Option	**meaning**
-f	force move despite target file permissions

mkdir	**Create Specified Directories**
	mkdir [options] dirnames

Option	**meaning**
-m *mode*	specify directory’s mode 9 (see chmod)
-p	create parent directories that don’t already exist

passwd	**Change login Password**
	passwd

print prints *arguments* to stdout see echo

print [*options*] *arguments*

Sequence	Character printed (these are embedded in *arguments*)
\a	alert or <ctrl-G> (rings the bell)
\b	backspace or <ctrl-H>
\c	omit final newline
\f	form feed or <ctrl-L>
\n	newline or <ctrl-J>
\r	carriage return or <ctrl-M>
\t	tab or <ctrl-I>
\v	vertical tab or <ctrl-K>
\0*num*	ASCII character with octal (base 8) value *num*, where *num* 1-3 digits
\\	backslash

Option	Meaning
-n	omit final newline
-r	Raw; ignore the escape sequences listed above
-p	print on pipe to coroutine
-s	print to command history file
-u*n*	print to file descriptor *n*

ps Report Process Status

ps [*options*]

Option	Meaning
-a	print all processes except group leaders and non-terminal associated
-c *file*	use *file* for core image (/dev/mem default)
-d	print all processes except group leaders
-e	print all processes
-f	print full listing
-g *list*	list only processes whose leaders are in *list*
-l	long listing (more info than –f)
-n *list*	use *list* for namelist (/unix default)
-p *list*	list only processes whose ids are in *list*
-s *dev*	use *dev* for swap device (/dev/swap default)
-t *list*	list only processes of terminals in *list*
-u *list*	list only processes with user-IDs in *list*
list	comma or blank separated list with optional enclosing quotes

pwd Print Working Directory Name

pwd

read	**Read stdin and store in the specified variables**
	read [*options*] var1 var2 …
	read [options] var?"prompt string"

Option	**meaning**
-r	raw ; do not use \ as a line continuation character
-p	read from pipe to coroutine
-s	Save input in command history file
-u*n*	Read from file descriptor *n*

rm	**Remove Files**
	rm [*options*] *files*

Option	**meaning**
-f	force removal of files without write permission
-j	ask for confirmation before each delete
-r	recursively delete directories

rmdir	**Remove Empty Directories**
	rmdir [options] directories

Option	**meaning**
-p	remove empty parent directories
-s	suppress error messages

rsh	**Restricted Shell (See RM)**
	rsh [*options*] [*args*]

Option	**meaning**
-a	mark modified export variables
-c *cmd*	execute *cmd* (default reads commands from file named in first entry of *args*)
-e	if non-interactive, exit if a command fails
-f	disable wildcarding
-h	locate and remember functions on definition instead of on execution
-i	set interactive mode
-k	all keyword arguments placed in environment
-n	read commands without executing them
-r	set restricted mode
-s	read commands from stdin
-u	set error upon substituting an unset variable
-v	print input lines as read
-x	print commands, as executed, with arguments

sed Stream editor

sed [*0ptions*] [*files*] stdin read if no *files* specified
see awk and sed specific section at back

Option	**meaning**
-e *script*	editor commands in script executed
-f *file*	editor commands read from *file*
-n	suppress unrequested output

sort Sort/manage Files

sort [*options*] [*files*] stdin read if- or no *files* specified

Option	**Meaning**
-b	ignore leading tabs and spaces
-d	dictionary order (use only letters, digits, tabs and spaces)
-f	sort upper case and lower case letters together
-i	ignore non-printing characters in comparisons
-m	merge already sorted *files*
-M	compare as months (implies-b)
-n	numeric sort (implies-b)
-o *output*	place sorted results in *output*
-r	reverse sort; descending order
-t*c*	set field separator to *c* (tab default0
-u	output only one occurrence of duplicate lines
-y[*mem*]	Kbytes of memory to start with, 0 = minimum no arg = maximum
-z[*size*]	bytes in longest line read
+pos1 [-pos2]	sort only from *pos1* to *pos2*
	if *pos2* not specified, key includes up to the end of line
	pos1 and *pos2* of the form: *m*[.*n*][bdfinr]
	m *m* fields from start of line skipped (0 default)
	n *n* characters from start of field skipped (0 default)
	bdfinr option applies only to specified key

su Become Another User

su [*0ption*] [*user* [*args*]]

Option	Meaning
-	change environment as if user logged in

tail	**Output Last Part of File**	
	tail [*options*] [*file*]	stdin read if *file* not specified

Option	**Meaning**
-f	follow growth of file (don't stop at end of file)
+*n*[bcl]	begin *n* units from beginning of file, may be blocks, characters, or lines (default)
-*n*[bcl]	begin *n* units before end of file (10 default)

tar	**Tape File Archiver**	
	tar [*key*] [*files*]	stdin read if no *file*s specified
	/etc/tar-[*key*] [*files*]	Key Format: *letter* [modifier]

Option	**Meaning**
	Key Letters
c	create new tape and record *files* , implies r
r	record *files* onto end of tape
t	tell when *files* found, all entries if no *files*
u	update tapes by adding *files* if not on tape or if modified since being last written to tape extract *files,* entire tape if no *files*
	Key Modifiers
#ldensity	# is tape drive number (0..7), (0 default)
h	high (6250 bpi)
l	low (800 bpi)
m	medium (1600 bpi) (default)
b *n*	*n* is blocking facter (1default, 20 max)
f *arch*	*arch* is the file to be used for input/output to archives (if '–' then stdin read
l	complain if all file links not found
m	update file modification times
o	set user and group id of extracted files to user running tar
v	verbose mode
w	wait for confirmation after reporting filename (y causes action to be performed)

tee	**Copy stdn to stdin to stdout and Files**
	tee {*options*] {*files*]

Option	**Meaning**
-a	append to *files* instead of overwriting
-i	ignore interrupts

test	**Condition Evaluation**	
	test *expression*	Bourne shell compatible
	[expression]	see [[…]]

Option	**Meaning**
	Expressions
-b *file*	true if *file* exists and is a block special *file*
-c *file*	true if the *fil*e exists and is a character special *file*
-d *file*	true if the *file* exists and is a directory
-f *file*	true if *file* exists and is a regular *file*
-g *file*	true if *file* exists and has set-GID bit set
-k *file*	true if *file* exists and has sticky bit set
-n string	true if *string* is of non-zero length
n1-eq *n2*	true if integers *n1* and *n2* equal
n1-ge *n2*	true if integer *n1* ≥ and *n2*
n1 –gt *n2*	true if integer *n1* > *n2*
n1 –le *n2*	true if integer n1 ≤ *n2*
n1-lt *n2*	true if integer *n1* < *n2*
n1-ne *n2*	true if integers *n1* and *n2* unequal
-p *file*	true if *file* exists and is a named pipe
-r *file*	true if *file* exists and is readable
-s *file*	true if *file* exists and has non-zero size
string	true if *string* is not the null string
s1 = s2	true if strings s1 and *s2* are the same
s1 != *s2*	true if strings *s1* and *s2* are not the same
-t [*fd*]	true if descriptor *fd* associated with terminal
-u *file*	true if *file* exists and has set-UID bit set
-w *file*	true if *file* exists and is writable
-x *file*	true if *file* exists and is executable
-z *string*	true if *string* has zero length
	Expressions may be joined by:
!	logical negation
-a	logical and
-o	logical or
\\(*expr*\\)	grouping parentheses (escaped from shell)

touch	**Update File Access/Modification Times**	
	touch {*options*] *files*	the optional date format is: MMDDhhmm[yy]

Option	**Meaning**
-a [*date*]	update only access time
-c	do not create non-existent *files*
-m [*date*]	update only modification time

tr Translate Characters

tr [options] [*string1* [*string2*]]

Option	Meaning
-c	complement *string*1 with 001-377 (octal)
-d	delete characters in *string1* from input
-s	squeeze repeated output characters in *string2*

Strings may include

[*a*-*z*]	short form for range of characters from a to *z*
[*a***n*]	short form for *n* repetitions of character *a*

typeset Control variable behaviour

Option	Meaning
	With no option, create local variable within function.
-L	Left justify and remove leading blanks.
-R	Right justify and remove trailing blanks.
-f	With no arguments, prints all function definitions.
-f *fname*	Prints the definition of function *fname*.
+f	Prints all function names.
-ft	Turns on trace mode for named function(s)
+ft	Turns off trace mode for named function(s)
-fu	Defines given name(s) as auto-loaded function(s)
-i*n*	declare variable as an integer. The optional *n* sets the base for display
-l	Convert all letters to lowercase.
-r	Make variable read-only.
-u	Convert all letters to uppercase.
-x	Export variable, i.e. put in environment and pass to sub-shells.

umask Set File Creation Mask

umask [*option*]	if option not specified, current mask printed

Option	Meaning
ugo	3 digit code specifying denied file access permissions. Each of the *ugo* digits formed of *read* (04), *write* (02), & *execute* (01) permissions for the classifications of user, group, & others.

unset	**delete variable of function from memory**
	usage [options] [*name*]

Option	**Meaning**
-f	name refers to a function

usage	**Command Usage Examples**	
	usage [options] [*command*]	menu displayed if no arguments given

Option	**Meaning**
-d	print command description
-e	print command examples
-o	print command options
command	a UNIX command to print info for

vi	**Screen Editor**
	vi [*options*] [*files*]

Option	**Meaning**
=pos	position file at *pos* (end of file default)
-l	set options appropriately for editing LISP
-r	retrieve last saved version of *file* after system or editor crash (list of all saved files default)
-R	read-only mode (same as view)
t *tag*	edit file containing *tag* and position editor at its definition
-w*n*	set default window size to *n*
-x	create or edit encrypted file
pos	any editor command not containing a space

view	**Read –Only Screen Editor**
	view [options] [files]

Option	**Meaning**
+*pos*	position file at *pos* (end of line default)
-l	set options appropriately for editing LISP
-r	retrieve last saved version of *file* after system or editor crash (list of all saved files default)
-t *tag*	edit file containing tag and position editor at its definition
-w*n*	set default window size to *n*
-x	create or edit encrypted file
pos	any editor command not containing a space

wait **Wait for All Background Processes to Complete**

wait [*id*]

Option	**Meaning**
id	process id to wait for

wc **Count lines, Words and Characters**

wc [*options*] [*files*] stdin read if no *files* specified

Option	**Meaning**
-c	output character counts
-l	output line counts
-w	output word counts

who **Who is on the System**

who [*options*] [*file*] [am i]

Option	**Meaning**
-a	turn all options on
-b	list time and date of last reboot
-d	list expired processes not respawned by init
-l	list lines available for login
-p	list active processes spawned by init
-r	list info on run-level of init processes
-s	list current user’s name, line and time logged in (default)
-t	show last time date changed clock
-t	list info on state of terminal
-u	long list of info on logged in users
file	read instead of /et/utmp for login information
am i	outputs who you are logged in as

Awk System variables

Variable	Description
FILENAME	Current filename
FS	Field Separator (defaults to blank)
NF	Number of fields in current record
NR	Number of the current record
OFS	Output Field Separator (defaults to blank)
ORS	Output Record Separator (defaults to newline)
RS	Record Separator (defaults to newline)
nawk	
ARGC	Number of arguments on command line
ARGV	An array containing the command line arguments
FNR	Like NR but relative to the current file
CONVFMT	Output format for numbers (defaults to %.6g)
RSTART	First position in string matched by the match function
RLENGTH	Length of the string matched by the match function
SUBSEP	Separator character for array subscripts (defaults to 0x1C)
gawk	
ENVIRON	An associative array of environment variables
IGNORECASE	0 indicates case-sensitive pattern matching. Non-zero value then case is ignored. (Applies to all characters including $FS)

Escape sequences

Sequence	Description
\a	Alert character (usually the bell character - <ctrl-G>)
\b	backspace
\f	form feed
\n	newline
\r	carriage return
\t	Horizontal tab
\v	Vertical tab
\c	any literal character c (eg \\\\ is a backslash while \\” is a double quote)
\0*ddd*	1 to 3 digit octal value
\x*hex*	Character represented as a hexadecimal character

Arithmetic Operators

Operator	Description
^	Exponentiation
*	Multiplication
/	Division
%	Modulo
+	Addition
-	Subtraction

Assignment Operators

Operator	Description
=	Standard assignment
+=	Assign result of addition
-=	Assign result of subtraction
*=	Assign result of multiplication
/=	Assign result of division
%=	Assign result of modulo
++	Add 1 to variable
--	Subtract 1 from variable

Relational Operators

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to
~	Matches
!~	Does not match
||	Logical OR
&&	Logical AND

Format Specifiers Used in printf

Character	**Description**
c	ASCII character
d	Decimal integer
e	Floating point format ([-]*d.precisionE*[+-]*dd*).
f	Floating point format ([-]*ddd.precision*).
g	shorter of e or f conversion, with trailing zeroes removed.
o	Unsigned octal value.
s	String
x	Unsigned hexadecimal number.
%	Literal %

Awk's Built-in Arithmetic Functions

Awk Function	**Description**
*cos(*x*)*	Returns cosine of x in radians.
exp(*x*)	Returns exponent of x.
int(*x*)	Returns truncated value of x
log(*x*)	Returns logarithm of x.
sin(*x*)	Returns sine of x in radians.
sqrt(*x*)	Returns square root of x
Nawk Function	**Description**
atan2(*y*,*x*)	Returns arctangent of y/x in the range -π to π.
rand( )	Returns pseudo-random number *r*, where 0<= r <1.
srand(*x*)	Establishes new seed for rand( ). If no seed is specified, uses time of day.

Awk's Built-in String Functions

Awk Function	**Description**
index(s,t*)*	Returns position of substring *t* in string *s* or zero if not present.
length(*s*)	Returns length of string *s*.
split(*s*,*a*,*sep*)	Parses string *s* into elements of array a using field separator *sep*; returns number of elements. If *sep* is not supplied, FS is used.
sprintf("*fmt",expr*)	Uses printf format specification for expr.
substr(*s*,*p*,*n*)	Returns substring of string *s* at beginning position *p* up to a maximum length of *n*. If *n* is not supplied, the rest of the string from *p* is used.
Nawk Function	
gsub(*r*,*s*,*t*)	Globally substitutes s for each match of the regular expression *r* in the string *t*. Returns the number of substitutions. If *t* is not supplied, defaults to#0.
match(*s*,*r*)	Returns either the position in *s* where the regular expression *r* begins, or 0 if no occurrences are found. Sets the value of RSTART and RLENGTH.
sub(*r*,*s*,*t*)	Substitutes *s* for first match of the regular expression *r* in the string *t*. Return 1 if successful; 0 otherwise. If *t* is not supplied, defaults to $0.
Gawk function	
tolower(*s*)	Translates all uppercase characters in string *s* to lowercase and returns the new string.
toupper(s)	Translates all lowercase characters in string *s* to uppercase and returns the new string.

Summery of Metacharacters

Special Characters	Usage
.	Matches any *single* character except *newline*.
*	Matches any number (including zero) of the single character (including a character specified by a regular expression) that immediately precedes it.
[…]	Matches any *one* of the class of characters enclosed between the brackets. A circumflex (^) as first character inside brackets reverses the match to all characters except newline and those listed in the class. A hyphen (-) is used to indicate a range of characters. The close bracket (]) and hyphen as the first character in class is a member of the class. All other meta-characters lose their meaning when specified as members of a class.
^	First character of regular expression, matches the beginning of the line.
\{n,m\}	Matches a range of occurrences of the single character (including a character specified by a regular expression) that immediately precedes it. \{*n*\} will match exactly n occurrences, \{*n*,\} will match at least *n* occurrences, and \{*n,m*\} will match any number of occurrences between *n* and *m*. (sed and grep only.)
$	As last character of regular expression, matches the end of the line.
\	Escapes the special character that follows.

Extended Metacharacters (egrep and awk)

Special Characters	Usage
+	Matches one or more occurrences of the preceding regular expression.
?	Matches zero or one occurrences of the preceding regular expression.
( )	Grpups regular expressions.
|	Specifies that either the preceding or following regular expression can be matched.

Special Filenames: Gawk

File name	Description
/dev/stdin	Standard input
/dev/stdout	standard output
/dev/stderr	Standard error.
/dev/fd*n*	The file referenced as file descriptor *n*.