Special Characters

&	Background job
#	Comment
~	Home Directory
!	Logical NOT
‘	Quote (Strong)
“	Quote (Weak)
<	Redirect input
>	Redirect output
>>	Redirect output + append to file
|	Redirect (pipe) output to next command
/	Separator for pathname directories
;	Separator for shell commands
[ ]	Start and end a character-set wildcard
{ }	Start and end a command block
( )	Start and end a subshell
(( ))	Perform arithmetic
*****	Wildcard
?	Wildcard – single character
$	Variable expression
****	Escape a special character
n>&m	Descriptor n is a copy of ouput file descriptor m
n<&m	Descriptor n is a copy of input file descriptor m

String Operators

${varname:-word}	Returns word
${varname:=word}	Sets and returns word
${varname:?message}	Prints message and exits
${varname:offset:length}	Returns substring
${varname:+word}	If varname is defined, return word

Pattern-matching operators

${varname#pattern}	Match first from the start
${varname##pattern }	Match last from the start
${varname%pattern}	Match first from the end
${varname/pattern/replace}	Match longest and replace
${varname//pattern/replace}	Match all and replace

Variables

$0, $1, $2,	Positional parameters
$@	“$1” “$2” “$3” …
$*	A string of positional params > 0
$#	Number of positional params
$?	Exit status of last command run

Functions

define	function myfunction { … } *or* myfunction ( ) { … }
call	myfunction arg1 arg2 …
keywords	local – limit var scope

If / else conditions

x && y	If x runs, then run y
x || y	If x fails, then run y
x -a y	x AND y
x -o y	x OR y
-lt, -le, -eq, -gt, -ge, -ne	Integer comparisons
=, !=, <, >	String comparisons
-n str1	str1 has length > 0 (nonzero)
-z str1	str1 has length 0 (zero)
-d file	File exists and is a directory
-e file	File exists
-f file	File exists and is a regular file
-r file	User has read permission on file
-s file	File exists and is non empty
-w file	User has write permission on file
-x file	User has execute permission on file, or search if directory
-N file	File was modified since it was left read
-O file	User owns file
-G file	File’s group ID matches the user’s group ID
file1 -nt file2	file1 has newer modification time than file2

Flow control sentences

if	if *condition*; then *commands*; fi
for	for ((*init*; *condition*; *increment*)); do *commands*; done
for	for *var* in *array*; do *commands*; done
case	case *expression* in *pattern1*) *commands* ;; *pattern2*) *commands* ;; *) *commands* ;; esac
while	while *condition*; do *commands*; done
until	until *condition*; do *commands*; done

Arrays

Arr_name=(‘el1’ ‘ el2’ ‘el3’)	define
Arr_name[index]	Element #index
Arr_name[-1]	Last element
Arr_name[@]	All elements, space-separated
#Arr_name[@]	Array length
#Arr_name[index]	String length of the Nth element
Arr_name[@]:m:n	Range (from position m, length n
!Arr_name[@]	Keys of all elements
Arr_name=("${Arr_name[@]}” “newElement”)	Push
Arr_name+=('newElement')	Also Push
unset Arr_name[n]	Remove one item

Dictionaries

declare –A dict	Define
dict[key]=”value”	Define value of a key
dict[key]	Value of a key
dict[@a]	All values
!dict[@]	All keys
#dict[@]	Number of elements
unset dict[key]	Delete the key

Useful Commands

type <cmd>	Determine type of command: -a ; displays all the locations
builtin <cmd>	Run builtin commands explicitly
which <cmd>	Locate the executable of a command: -a ; show all locations
clear	Clear the terminal screen
echo “str1”	Print message to terminal screen: -e ; uses escape sequences like (\n = newline, \t = tab) -n ; supresses automatic newline after print
printf <format> <variables>	Print messages to terminal screen. Formatting be like: %s – String %-Xs – String wide X chars, left aligned %Xs – String wide X chars, right aligned %d – Integer (%-Xd, %Xd) %f – Float %.Xf - Round to X decimal spaces
date <options> <+format>	Will display date and time. Formats (“+%Y-%m-%d”): %Y – Year, %m – month, %d – day, %H – hours, %M – minutes, %S – seconds, (%A uppercase for full name) %a – DayOfTheWeek, (%B) %b - Month Options (-d “yesterday”): “yesterday”, “next Monday”,
read <options> <variable>	Read input from user or file and store into variable (read var1). Options: -p “Text” : print before input -a : store the input in array
history <options>	Display the command history for that session. Options: -c : clear the history -X : print the last X commands -a : appends history to bash history file -d X : deletes the command with index X from history
sleep <num_time>	Delay the execution of a script. Num_time: Xs : delay for X second(s) (default) Xm : delay for X minute(s) Xh : delay for X hour(s)
man <command>	Opens the manual pages for the <command>.
ls <options> <path>	List the files and directories in the current working directory or given path. Options: -l : list detailed view for files -a : show all files, even hidden -alp : ???
pwd	Display the current working directory.
cd <directory>	Change the current working directory. <directory>: ‘/path’ : changes directory to path ‘..’ : changes to parent directory of the current one ‘~username’ : changes to home directory for username ‘-’ : changes to previous working directory used
mkdir <directory>	Creates new directory. <directory> can be: ‘d1’ : creates new directory called d1 ‘d1’ ‘d2’ ‘d3’ : creates more directories in the current one -p ‘d1/d2’ : creates d1 and another directory d2 as d1’s child
rmdir <directory>	Works the same as mkdir, but it deletes the directory if it is empty.
cat <file>	Display the contents of the file on the terminal. <file>: ‘file.txt’ : displays file.txt ‘f1.txt’ ‘f2.txt’ : displays files consecutively -n ‘file.txt’ : displays file.txt with numbered lines
more, less, od, hexdump	More and less are both text viewers, od gives octal output and hexdump hexadecimal.
vi, vim, emacs, nano	File editors. Use ‘man file_editor’ to learn how to use them.
cp <source> <destination>	Copy files or directories from source to destination. cp file /path : copy file to path cp –r directory /path : copy directory with all its contents to path
mv <source> <destination>	Moves files or directories from source to destination. mv file /path : move file to path mv directory /path : move directory to path mv file.txt newfile.txt : renames file.txt to newfile.txt
uniq <options> <file>	Removes all consecutive lines. Options: -c : also counts the amount of duplicates -i : ignores the case -d : outputs only duplicates -u : outputs only the unique

rev <file>	Reverse the characters in each line of the input stream or file
tr <options> <set1> <set2> <file>	Translate or delete characters. Set1 is translated to Set2. -d : removes the characters -c : complement the Set1
wc <options> <file>	Counts the number of lines, words, bytes. Options: -l : only counts the lines -w : only counts the words -c : only counts the bytes
grep <options> <pattern> <file>	Search for specific pattern or regular expression. Options: -i : ignore case -v : invert the match (print only the lines not matching the pattern) -w : match only whole words -n : print the line numbers for each match -r : search recursively through directories
shift <X>	Shift the positional parameters to the left. X is number of positions to shift.
jobs <options>	Display a list of jobs that are currently running in the background or are suspended. -l : also displays PID of a job -p : displays only the PIDs -r : displays the running jobs -s : displays the stopped jobs
fg <JID>	Bring a job that is running in the background to the foreground.
bg <JID>	Start a suspended job in the background.
disown %<JID>	Remove jobs from shell's job control. (disown %2 : removes job with JID 2)
ulimit <options>	Display the resource limits of the current shell and its children. -a : displas all current limits