|  |
| --- |
| #!/bin/bash |
|  | ############################################################################## |
|  | # SHORTCUTS and HISTORY |
|  | ############################################################################## |
|  |  |
|  | CTRL+A # move to beginning of line |
|  | CTRL+B # moves backward one character |
|  | CTRL+C # halts the current command |
|  | CTRL+D # deletes one character backward or logs out of current session, similar to exit |
|  | CTRL+E # moves to end of line |
|  | CTRL+F # moves forward one character |
|  | CTRL+G # aborts the current editing command and ring the terminal bell |
|  | CTRL+H # deletes one character under cursor (same as DELETE) |
|  | CTRL+J # same as RETURN |
|  | CTRL+K # deletes (kill) forward to end of line |
|  | CTRL+L # clears screen and redisplay the line |
|  | CTRL+M # same as RETURN |
|  | CTRL+N # next line in command history |
|  | CTRL+O # same as RETURN, then displays next line in history file |
|  | CTRL+P # previous line in command history |
|  | CTRL+R # searches backward |
|  | CTRL+S # searches forward |
|  | CTRL+T # transposes two characters |
|  | CTRL+U # kills backward from point to the beginning of line |
|  | CTRL+V # makes the next character typed verbatim |
|  | CTRL+W # kills the word behind the cursor |
|  | CTRL+X # lists the possible filename completions of the current word |
|  | CTRL+Y # retrieves (yank) last item killed |
|  | CTRL+Z # stops the current command, resume with fg in the foreground or bg in the background |
|  |  |
|  | ALT+B # moves backward one word |
|  | ALT+D # deletes next word |
|  | ALT+F # moves forward one word |
|  | ALT+H # deletes one character backward |
|  |  |
|  | BACKSPACE # deletes one character backward |
|  | DELETE # deletes one character under cursor |
|  |  |
|  | history # shows command line history |
|  | !! # repeats the last command |
|  | !<n> # refers to command line 'n' |
|  | !<string> # refers to command starting with 'string' |
|  |  |
|  | exit # logs out of current session |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # BASH BASICS |
|  | ############################################################################## |
|  |  |
|  | env # displays all environment variables |
|  |  |
|  | echo $SHELL # displays the shell you're using |
|  | echo $BASH\_VERSION # displays bash version |
|  |  |
|  | bash # if you want to use bash (type exit to go back to your previously opened shell) |
|  | whereis bash # locates the binary, source and manual-page for a command |
|  | which bash # finds out which program is executed as 'bash' (default: /bin/bash, can change across environments) |
|  |  |
|  | clear # clears content on window (hide displayed lines) |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # FILE COMMANDS |
|  | ############################################################################## |
|  |  |
|  |  |
|  | ls # lists your files in current directory, ls <dir> to print files in a specific directory |
|  | ls -l # lists your files in 'long format', which contains the exact size of the file, who owns the file and who has the right to look at it, and when it was last modified |
|  | ls -a # lists all files in 'long format', including hidden files (name beginning with '.') |
|  | ln -s <filename> <link> # creates symbolic link to file |
|  | touch <filename> # creates or updates (edit) your file |
|  | cat <filename> # prints file raw content (will not be interpreted) |
|  | any\_command > <filename> # '>' is used to perform redirections, it will set any\_command's stdout to file instead of "real stdout" (generally /dev/stdout) |
|  | more <filename> # shows the first part of a file (move with space and type q to quit) |
|  | head <filename> # outputs the first lines of file (default: 10 lines) |
|  | tail <filename> # outputs the last lines of file (useful with -f option) (default: 10 lines) |
|  | vim <filename> # opens a file in VIM (VI iMproved) text editor, will create it if it doesn't exist |
|  | mv <filename1> <dest> # moves a file to destination, behavior will change based on 'dest' type (dir: file is placed into dir; file: file will replace dest (tip: useful for renaming)) |
|  | cp <filename1> <dest> # copies a file |
|  | rm <filename> # removes a file |
|  | find . -name <name> <type> # searches for a file or a directory in the current directory and all its sub-directories by its name |
|  | diff <filename1> <filename2> # compares files, and shows where they differ |
|  | wc <filename> # tells you how many lines, words and characters there are in a file. Use -lwc (lines, word, character) to ouput only 1 of those informations |
|  | sort <filename> # sorts the contents of a text file line by line in alphabetical order, use -n for numeric sort and -r for reversing order. |
|  | sort -t -k <filename> # sorts the contents on specific sort key field starting from 1, using the field separator t. |
|  | chmod -options <filename> # lets you change the read, write, and execute permissions on your files (more infos: SUID, GUID) |
|  | gzip <filename> # compresses files using gzip algorithm |
|  | gunzip <filename> # uncompresses files compressed by gzip |
|  | gzcat <filename> # lets you look at gzipped file without actually having to gunzip it |
|  | lpr <filename> # prints the file |
|  | lpq # checks out the printer queue |
|  | lprm <jobnumber> # removes something from the printer queue |
|  | genscript # converts plain text files into postscript for printing and gives you some options for formatting |
|  | dvips <filename> # prints .dvi files (i.e. files produced by LaTeX) |
|  | grep <pattern> <filenames> # looks for the string in the files |
|  | grep -r <pattern> <dir> # search recursively for pattern in directory |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # DIRECTORY COMMANDS |
|  | ############################################################################## |
|  |  |
|  |  |
|  | mkdir <dirname> # makes a new directory |
|  | rmdir <dirname> # remove an empty directory |
|  | rmdir -rf <dirname> # remove a non-empty directory |
|  | mv <dir1> <dir2> # rename a directory from <dir1> to <dir2> |
|  | cd # changes to home |
|  | cd .. # changes to the parent directory |
|  | cd <dirname> # changes directory |
|  | cp -r <dir1> <dir2> # copy <dir1> into <dir2> including sub-directories |
|  | pwd # tells you where you currently are |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # SSH, SYSTEM INFO & NETWORK COMMANDS |
|  | ############################################################################## |
|  |  |
|  |  |
|  | ssh user@host # connects to host as user |
|  | ssh -p <port> user@host # connects to host on specified port as user |
|  | ssh-copy-id user@host # adds your ssh key to host for user to enable a keyed or passwordless login |
|  |  |
|  | whoami # returns your username |
|  | passwd # lets you change your password |
|  | quota -v # shows what your disk quota is |
|  | date # shows the current date and time |
|  | cal # shows the month's calendar |
|  | uptime # shows current uptime |
|  | w # displays whois online |
|  | finger <user> # displays information about user |
|  | uname -a # shows kernel information |
|  | man <command> # shows the manual for specified command |
|  | df # shows disk usage |
|  | du <filename> # shows the disk usage of the files and directories in filename (du -s give only a total) |
|  | last <yourUsername> # lists your last logins |
|  | ps -u yourusername # lists your processes |
|  | kill <PID> # kills the processes with the ID you gave |
|  | killall <processname> # kill all processes with the name |
|  | top # displays your currently active processes |
|  | bg # lists stopped or background jobs ; resume a stopped job in the background |
|  | fg # brings the most recent job in the foreground |
|  | fg <job> # brings job to the foreground |
|  |  |
|  | ping <host> # pings host and outputs results |
|  | whois <domain> # gets whois information for domain |
|  | dig <domain> # gets DNS information for domain |
|  | dig -x <host> # reverses lookup host |
|  | wget <file> # downloads file |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # VARIABLES |
|  | ############################################################################## |
|  |  |
|  |  |
|  | varname=value # defines a variable |
|  | varname=value command # defines a variable to be in the environment of a particular subprocess |
|  | echo $varname # checks a variable's value |
|  | echo $$ # prints process ID of the current shell |
|  | echo $! # prints process ID of the most recently invoked background job |
|  | echo $? # displays the exit status of the last command |
|  | read <varname> # reads a string from the input and assigns it to a variable |
|  | let <varname> = <equation> # performs mathematical calculation using operators like +, -, \*, /, % |
|  | export VARNAME=value # defines an environment variable (will be available in subprocesses) |
|  |  |
|  | array[0]=valA # how to define an array |
|  | array[1]=valB |
|  | array[2]=valC |
|  | array=([2]=valC [0]=valA [1]=valB) # another way |
|  | array=(valA valB valC) # and another |
|  |  |
|  | ${array[i]} # displays array's value for this index. If no index is supplied, array element 0 is assumed |
|  | ${#array[i]} # to find out the length of any element in the array |
|  | ${#array[@]} # to find out how many values there are in the array |
|  |  |
|  | declare -a # the variables are treated as arrays |
|  | declare -f # uses function names only |
|  | declare -F # displays function names without definitions |
|  | declare -i # the variables are treated as integers |
|  | declare -r # makes the variables read-only |
|  | declare -x # marks the variables for export via the environment |
|  |  |
|  | ${varname:-word} # if varname exists and isn't null, return its value; otherwise return word |
|  | ${varname:=word} # if varname exists and isn't null, return its value; otherwise set it word and then return its value |
|  | ${varname:?message} # if varname exists and isn't null, return its value; otherwise print varname, followed by message and abort the current command or script |
|  | ${varname:+word} # if varname exists and isn't null, return word; otherwise return null |
|  | ${varname:offset:length} # performs substring expansion. It returns the substring of $varname starting at offset and up to length characters |
|  |  |
|  | ${variable#pattern} # if the pattern matches the beginning of the variable's value, delete the shortest part that matches and return the rest |
|  | ${variable##pattern} # if the pattern matches the beginning of the variable's value, delete the longest part that matches and return the rest |
|  | ${variable%pattern} # if the pattern matches the end of the variable's value, delete the shortest part that matches and return the rest |
|  | ${variable%%pattern} # if the pattern matches the end of the variable's value, delete the longest part that matches and return the rest |
|  | ${variable/pattern/string} # the longest match to pattern in variable is replaced by string. Only the first match is replaced |
|  | ${variable//pattern/string} # the longest match to pattern in variable is replaced by string. All matches are replaced |
|  |  |
|  | ${#varname} # returns the length of the value of the variable as a character string |
|  |  |
|  | \*(patternlist) # matches zero or more occurrences of the given patterns |
|  | +(patternlist) # matches one or more occurrences of the given patterns |
|  | ?(patternlist) # matches zero or one occurrence of the given patterns |
|  | @(patternlist) # matches exactly one of the given patterns |
|  | !(patternlist) # matches anything except one of the given patterns |
|  |  |
|  | $(UNIX command) # command substitution: runs the command and returns standard output |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # FUNCTIONS |
|  | ############################################################################## |
|  |  |
|  |  |
|  | # The function refers to passed arguments by position (as if they were positional parameters), that is, $1, $2, and so forth. |
|  | # $@ is equal to "$1" "$2"... "$N", where N is the number of positional parameters. $# holds the number of positional parameters. |
|  |  |
|  |  |
|  | function functname() { |
|  | shell commands |
|  | } |
|  |  |
|  | unset -f functname # deletes a function definition |
|  | declare -f # displays all defined functions in your login session |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # FLOW CONTROLS |
|  | ############################################################################## |
|  |  |
|  |  |
|  | statement1 && statement2 # and operator |
|  | statement1 || statement2 # or operator |
|  |  |
|  | -a # and operator inside a test conditional expression |
|  | -o # or operator inside a test conditional expression |
|  |  |
|  | # STRINGS |
|  |  |
|  | str1 == str2 # str1 matches str2 |
|  | str1 != str2 # str1 does not match str2 |
|  | str1 < str2 # str1 is less than str2 (alphabetically) |
|  | str1 > str2 # str1 is greater than str2 (alphabetically) |
|  | str1 \> str2 # str1 is sorted after str2 |
|  | str1 \< str2 # str1 is sorted before str2 |
|  | -n str1 # str1 is not null (has length greater than 0) |
|  | -z str1 # str1 is null (has length 0) |
|  |  |
|  | # FILES |
|  |  |
|  | -a file # file exists or its compilation is successful |
|  | -d file # file exists and is a directory |
|  | -e file # file exists; same -a |
|  | -f file # file exists and is a regular file (i.e., not a directory or other special type of file) |
|  | -r file # you have read permission |
|  | -s file # file exists and is not empty |
|  | -w file # your have write permission |
|  | -x file # you have execute permission on file, or directory search permission if it is a directory |
|  | -N file # file was modified since it was last read |
|  | -O file # you own file |
|  | -G file # file's group ID matches yours (or one of yours, if you are in multiple groups) |
|  | file1 -nt file2 # file1 is newer than file2 |
|  | file1 -ot file2 # file1 is older than file2 |
|  |  |
|  | # NUMBERS |
|  |  |
|  | -lt # less than |
|  | -le # less than or equal |
|  | -eq # equal |
|  | -ge # greater than or equal |
|  | -gt # greater than |
|  | -ne # not equal |
|  |  |
|  | if condition |
|  | then |
|  | statements |
|  | [elif condition |
|  | then statements...] |
|  | [else |
|  | statements] |
|  | fi |
|  |  |
|  | for x in {1..10} |
|  | do |
|  | statements |
|  | done |
|  |  |
|  | for name [in list] |
|  | do |
|  | statements that can use $name |
|  | done |
|  |  |
|  | for (( initialisation ; ending condition ; update )) |
|  | do |
|  | statements... |
|  | done |
|  |  |
|  | case expression in |
|  | pattern1 ) |
|  | statements ;; |
|  | pattern2 ) |
|  | statements ;; |
|  | esac |
|  |  |
|  | select name [in list] |
|  | do |
|  | statements that can use $name |
|  | done |
|  |  |
|  | while condition; do |
|  | statements |
|  | done |
|  |  |
|  | until condition; do |
|  | statements |
|  | done |
|  |  |
|  | ############################################################################## |
|  | # COMMAND-LINE PROCESSING CYCLE |
|  | ############################################################################## |
|  |  |
|  |  |
|  | # The default order for command lookup is functions, followed by built-ins, with scripts and executables last. |
|  | # There are three built-ins that you can use to override this order: `command`, `builtin` and `enable`. |
|  |  |
|  | command # removes alias and function lookup. Only built-ins and commands found in the search path are executed |
|  | builtin # looks up only built-in commands, ignoring functions and commands found in PATH |
|  | enable # enables and disables shell built-ins |
|  |  |
|  | eval # takes arguments and run them through the command-line processing steps all over again |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # INPUT/OUTPUT REDIRECTORS |
|  | ############################################################################## |
|  |  |
|  |  |
|  | cmd1|cmd2 # pipe; takes standard output of cmd1 as standard input to cmd2 |
|  | < file # takes standard input from file |
|  | > file # directs standard output to file |
|  | >> file # directs standard output to file; append to file if it already exists |
|  | >|file # forces standard output to file even if noclobber is set |
|  | n>|file # forces output to file from file descriptor n even if noclobber is set |
|  | <> file # uses file as both standard input and standard output |
|  | n<>file # uses file as both input and output for file descriptor n |
|  | n>file # directs file descriptor n to file |
|  | n<file # takes file descriptor n from file |
|  | n>>file # directs file description n to file; append to file if it already exists |
|  | n>& # duplicates standard output to file descriptor n |
|  | n<& # duplicates standard input from file descriptor n |
|  | n>&m # file descriptor n is made to be a copy of the output file descriptor |
|  | n<&m # file descriptor n is made to be a copy of the input file descriptor |
|  | &>file # directs standard output and standard error to file |
|  | <&- # closes the standard input |
|  | >&- # closes the standard output |
|  | n>&- # closes the ouput from file descriptor n |
|  | n<&- # closes the input from file descripor n |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # PROCESS HANDLING |
|  | ############################################################################## |
|  |  |
|  |  |
|  | # To suspend a job, type CTRL+Z while it is running. You can also suspend a job with CTRL+Y. |
|  | # This is slightly different from CTRL+Z in that the process is only stopped when it attempts to read input from terminal. |
|  | # Of course, to interrupt a job, type CTRL+C. |
|  |  |
|  | myCommand & # runs job in the background and prompts back the shell |
|  |  |
|  | jobs # lists all jobs (use with -l to see associated PID) |
|  |  |
|  | fg # brings a background job into the foreground |
|  | fg %+ # brings most recently invoked background job |
|  | fg %- # brings second most recently invoked background job |
|  | fg %N # brings job number N |
|  | fg %string # brings job whose command begins with string |
|  | fg %?string # brings job whose command contains string |
|  |  |
|  | kill -l # returns a list of all signals on the system, by name and number |
|  | kill PID # terminates process with specified PID |
|  | kill -s SIGKILL 4500 # sends a signal to force or terminate the process |
|  | kill -15 913 # Ending PID 913 process with signal 15 (TERM) |
|  | kill %1 # Where %1 is the number of job as read from 'jobs' command. |
|  |  |
|  | ps # prints a line of information about the current running login shell and any processes running under it |
|  | ps -a # selects all processes with a tty except session leaders |
|  |  |
|  | trap cmd sig1 sig2 # executes a command when a signal is received by the script |
|  | trap "" sig1 sig2 # ignores that signals |
|  | trap - sig1 sig2 # resets the action taken when the signal is received to the default |
|  |  |
|  | disown <PID|JID> # removes the process from the list of jobs |
|  |  |
|  | wait # waits until all background jobs have finished |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # TIPS & TRICKS |
|  | ############################################################################## |
|  |  |
|  |  |
|  | # set an alias |
|  | cd; nano .bash\_profile |
|  | > alias gentlenode='ssh admin@gentlenode.com -p 3404' # add your alias in .bash\_profile |
|  |  |
|  | # to quickly go to a specific directory |
|  | cd; nano .bashrc |
|  | > shopt -s cdable\_vars |
|  | > export websites="/Users/mac/Documents/websites" |
|  |  |
|  | source .bashrc |
|  | cd $websites |
|  |  |
|  |  |
|  | ############################################################################## |
|  | # DEBUGGING SHELL PROGRAMS |
|  | ############################################################################## |
|  |  |
|  |  |
|  | bash -n scriptname # don't run commands; check for syntax errors only |
|  | set -o noexec # alternative (set option in script) |
|  |  |
|  | bash -v scriptname # echo commands before running them |
|  | set -o verbose # alternative (set option in script) |
|  |  |
|  | bash -x scriptname # echo commands after command-line processing |
|  | set -o xtrace # alternative (set option in script) |
|  |  |
|  | trap 'echo $varname' EXIT # useful when you want to print out the values of variables at the point that your script exits |
|  |  |
|  | function errtrap { |
|  | es=$? |
|  | echo "ERROR line $1: Command exited with status $es." |
|  | } |
|  |  |
|  | trap 'errtrap $LINENO' ERR # is run whenever a command in the surrounding script or function exits with non-zero status |
|  |  |
|  | function dbgtrap { |
|  | echo "badvar is $badvar" |
|  | } |
|  |  |
|  | trap dbgtrap DEBUG # causes the trap code to be executed before every statement in a function or script |
|  | # ...section of code in which the problem occurs... |
|  | trap - DEBUG # turn off the DEBUG trap |
|  |  |
|  | function returntrap { |
|  | echo "A return occurred" |
|  | } |
|  |  |
|  | trap returntrap RETURN # is executed each time a shell function or a script executed with the . or source commands finishes executing |
|  |  |
|  | ############################################################################## |
|  | # COLORS AND BACKGROUNDS |
|  | ############################################################################## |
|  |  |
|  | # Reset |
|  | Color\_Off='\033[0m' # Text Reset |
|  |  |
|  | # Regular Colors |
|  | Black='\033[0;30m' # Black |
|  | Red='\033[0;31m' # Red |
|  | Green='\033[0;32m' # Green |
|  | Yellow='\033[0;33m' # Yellow |
|  | Blue='\033[0;34m' # Blue |
|  | Purple='\033[0;35m' # Purple |
|  | Cyan='\033[0;36m' # Cyan |
|  | White='\033[0;97m' # White |
|  |  |
|  | # Additional colors |
|  | LGrey='\033[0;37m' # Ligth Gray |
|  | DGrey='\033[0;90m' # Dark Gray |
|  | LRed='\033[0;91m' # Ligth Red |
|  | LGreen='\033[0;92m' # Ligth Green |
|  | LYellow='\033[0;93m'# Ligth Yellow |
|  | LBlue='\033[0;94m' # Ligth Blue |
|  | LPurple='\033[0;95m'# Light Purple |
|  | LCyan='\033[0;96m' # Ligth Cyan |
|  |  |
|  |  |
|  | # Bold |
|  | BBlack='\033[1;30m' # Black |
|  | BRed='\033[1;31m' # Red |
|  | BGreen='\033[1;32m' # Green |
|  | BYellow='\033[1;33m'# Yellow |
|  | BBlue='\033[1;34m' # Blue |
|  | BPurple='\033[1;35m'# Purple |
|  | BCyan='\033[1;36m' # Cyan |
|  | BWhite='\033[1;37m' # White |
|  |  |
|  | # Underline |
|  | UBlack='\033[4;30m' # Black |
|  | URed='\033[4;31m' # Red |
|  | UGreen='\033[4;32m' # Green |
|  | UYellow='\033[4;33m'# Yellow |
|  | UBlue='\033[4;34m' # Blue |
|  | UPurple='\033[4;35m'# Purple |
|  | UCyan='\033[4;36m' # Cyan |
|  | UWhite='\033[4;37m' # White |
|  |  |
|  | # Background |
|  | On\_Black='\033[40m' # Black |
|  | On\_Red='\033[41m' # Red |
|  | On\_Green='\033[42m' # Green |
|  | On\_Yellow='\033[43m'# Yellow |
|  | On\_Blue='\033[44m' # Blue |
|  | On\_Purple='\033[45m'# Purple |
|  | On\_Cyan='\033[46m' # Cyan |
|  | On\_White='\033[47m' # White |
|  |  |
|  | # Example of usage |
|  | echo -e "${Green}This is GREEN text${Color\_Off} and nornal text" |
|  | echo -e "${Red}${On\_White}This is Red test on White background${Color\_Off}" |
|  | # option -e is mandatory, it enable interpretation of backslash escapes |