Shell scripting tutorial… (Use Ubuntu)

BASH = Command line interpreter

|  |  |
| --- | --- |
| **Task** | **Code** |
| Changing to a windows directory… | * “\” is used as a space and will not actually be recognized as a backslash * C:\ drive is represented as /mnt/c/   **cd /mnt/c/Users/Peter/Documents/Bash\_Shell** |
| What if directory folder contains a space? | Cd /mnt/c/Users/Peter/Documents/Bash\_Shell/Exercise\ Files |
| Man [command]  Mkdir – Make directory folder  Rmdir – Remove directory  Rm -r [foldername] – remove directory (not working)  Cp [file] [new file.txt] – Make a new copy of a file  Cat [text.txt] – prints out text file  Head [file] – prints first couple lines of file. Also see tail. | Brings help page for command |
| **Shortcuts (Tilde and brace expansion)**  Brace expansion:  Touch {apple.txt,banana.txt,durian.txt}   * This creates 3 files simultaneously * Make sure there are NO spaces b/n filenames   Touch file\_{1..1000}  Rm \*   * Removes all files in directory   Tilde expansion:  Cd ~ : Changes back to original working directory | $ echo file{1..5..1}.txt  #from 1 to 5, incremented by 1  file1.txt file2.txt file3.txt file4.txt file5.txt |
| Piping: Taking the results of one command and using them as arguments for another command.   * [Some command with output] | function   Grep: search files with specific patterns of text  Grep [what you want to search] [name of file] |  |
| BASH Scripts:  #!/bin/bash  # comment | Greeting=‘hello’  Echo $Greeting, world \(planet\)  Echo “$Greeting” – prints out hello  Echo ‘$Greeting’ – prints out $Greeting |
| Nano my.sh – write script  Bash [script] – run your script  Alternatively,  Chmod +X filename – Make script executable  **./file\_name –** run your script | Install VIM (text editor)   * Sudo apt-get install vim * Sudo is a unix command; windows doesn’t have it * vim Test ; creates file called Test * i = INSERT (press esc first) * :w = SAVE * :wq = SAVE and QUIT * Display line number - : set number * : colorscheme elflord |
| Notes on variables:   * Make sure that when assigning variables, we leave no empty spaces.   Command substitution   * A=$(pwd) ; echo $A * Runs whatever in parenthesis, and then assigns this output to A.   Arithmetic expression   * A=$((12 + 123)) #Need double parenthesis for arithmetic * Ex: d=2 ; e=$((d + 2))   Printf -v d (passes output into a variable named d.) | Set attributes to variables   * Declare -i d=123 #set d to be an integer * Declare -r e=456 #set e to be a read only (i.e. can’t be modified arithmetically) * Declare -l f=”ONLYLOWER” #Set to be lower case * Declare -u g=”onlyupper” #Set to be upper case   Comparing operations: [[ ]]   * 0 = TRUE, 1 = FALSE * Note that: <, >, ==, ≤, etc are for STRINGS * For integers, we need to use: -lt, gt, -le, -ge, -eq, -ne   Ex: [[ 2 -lt 3 ]]; echo $?   * && = AND, || = OR * For strings; is.null = -z; is.not.null = -n |
| Notes on strings:  A=”hello”  B=”World”  C=$A$B => C=”helloWorld”   * Substrings:   D=${c:3} will give “loWorld”  D=${c:3:4} will give “loWo”  D=${c: -4} will give “orld” | Print length of string:   * Echo ${#a}   Edit string:   * ${variable/to\_replace/what\_to\_replace}   Ex: fruit=”apple banana banana”  Echo ${fruit/banana/durian} = apple durian banana  Echo ${fruit//banana/durian} = apple durian durian |
| Arrays = Lists   * A=c() #Empty array * A=(“apple” “banana” “cherry”) #No commas b/n elements * Echo ${b[2]} ; cherry #Subset starting from 0. * A[4]=”cal” #Add element to array * A+=(“mystring”) #append element to end of array. | * Echo ${A[@]} #Print the entire array * Echo ${A[@] : -1} #Print last element of array   Add keys to array   * Declare -A myarray   Myarray[color]=blue  Myarray[player]=Crosby  echo ${myarray["player"]}; crosby |
| Redirecting output:  Echo “some text” > file1.txt  Echo “This is what I want to direct” > [file to write]  [file to write] does not have to be pre-created, this command will create the file on its own. | Appending (add to the very last element):  Ex: Add text to the end of a textfile.  Echo “line to add” ≫ textfile  Copy contents of one file into another file:  Cat file1 >> file2  #copies content of file1 into a new file called file2 |
| If/else statement:  ((Integer comparison)) – not true actually  Just remember to use the -gt, -e,… for integers  If expression  Then  Do something  Elif expression; Then  Do something else  Fi | #!/bin/bash  if [ -e readme.txt]; then  echo 'The file "readme.txt" exists!'  else  echo 'The file "readme.txt does not exist.'  Fi  See **Loops\_Tutorial.sh** |
| While loop:  i=0  while [ $i -le 10 ]; do #Note the space between while and [  echo i:$i  ((i+=1))  done | For loop  For I in 1 2 3 # I in {1..100}  Do  Echo $i  done  Arrays:  Myarray=(“app” “ban “che”)  For I in ${myarray[@]}  Do  Echo $i  done |
| The \* notation will keep track of all files of the form specified.  \*.R  I.e. echo \*.R will print all R file names in the directory!  X=\*.R  Echo $X | Functions:  Function myfunction {  Do something  Echo $1 #$1 is the first argument passed to this function  }  Myfunction argument1 |
| Run an R script | First, change directory to where you have your R script  Rscript helloworld.R  Note: Using R CMD BATCH outputs a file called helloworld.r.ROUT |
| Change file name | Touch file #create a file named “file”  Mv [from] [to]  Ex: mv file file.txt #Change file name from file to file.txt |
| Move to new directory | Mv [filename] [absolute new directory path] |
|  |  |
|  | file="$1" #Save the 1st argument as "$file"  shift #drop the first argument from "$@"  echo "$@" > "$file" #write the remaining #arguments to "$file"  ./result.txt “what I want to write in result.txt” |