Basics-

bash scripts are typically saved as “.sh”

must make the file executable

ex-> $chmod +x path/to/our/file/name.sh

if you want to make this available to every terminal session you need to add it to the “.bashrc” file

#add comments to your code

all files begin with #!/bin/bash

$./name.sh executes the file

Variables\_Scalar-

variable names can contain only alphanumeric characters (but cannot start with a number)

ex-> \_Var\_1

if a variable is marked with the command “readonly” it cannot be changed

ex-> readonly Variable

if a variable is marked with the command “unset” it is then deleted and unusable

ex->unset Variable

unset cannot be used on readonly variables

local - within a current instance of a shell

environmental- available to any child process of the shell

shell- required by the shell to function

Variables\_Arrray -

ex-> array\_name = (1, 2,3, .. 7)

${array\_name[index]} how to call on an array

all conditional expressions must be placed in square brackets with spaces around them

ex-> [ $a <= $b ]

-eq equal to

-ne not equal to

-gt greater than

-lt less than

-ge greater than or equal to

-le less than or equal to

Boolean\_Operators-

! logical negation

-o logical or

ex-> [ $a -lt 20 -o $b -gt 100] is true

-a logical and

String\_Operators-

=

!=

-z checks if the given operand has zero length

-n checks if the given operand has a non-zero length

str - checks to see if the string is empty, if it is it returns False

File\_Test\_Operators-

-b file checks to see if the file is block special

-c file checks to see if the the file is character special

-d file checks to see if the file is a directory

-f file checks to see if the file is and ordinary file

-g file checks to see if the file has the set group ID (SGID) bit set

-k file checks to see if the file has its sticky bit set

-p file checks if file is a named pipe

-t file checks if file descriptor is open and associated with a terminal

-u file checks to see if the file has it set user ID (SUID)bit set

-r file checks if the file is readable

-w file checks if the file is writeable

-x file checks if the file is executable

-s file checks if the file has a size greater than 0

-e file checks if a file exists

Class Notes

\_\_\_\_\_\_\_\_\_\_\_

curl cut.sh/command

looks up what a command does, gives examples. Easier to read the man pages

Hot Keys

——————-

| ctrl+n | goes down in the command history file |
| --- | --- |
| ctrl+p | goes up in command history file |
| !number | ! stands for history. If you look up the history and refer to it by its number it will re-run that command |
| ctrl+f or alt+f | right |
| ctrl+b or alt+b | left |
| ctrl+e | will take you to the end of the line you were typing on |
| ctrl+u | will erase the whole line |
| ctrl+l | clears the screen |

Man Pages

———————-

| G | moves to the end of the man page |
| --- | --- |
| g | moves to the start of the man page |
| n | moves between search results |
| /pattern | searches forward |
| ?pattern | searches backward |
| &pattern | displays only matching lines |

Order of Evaluation

———————-

1. Redirection
   1. ex-> | a pipe or other directors like < or <<
2. Alias
   1. if you type alias it will show you all known aliases
3. Parameter Expansion, command substitution, arithmetic expansion, and would quote removal
4. Shell function
5. Built-In commands
6. Hash tables
7. Path variables
8. Error {eq. command not found}

Brace Expansion

———————-

touch file{1, 5, 10}.log

will run that command for every file specified

Can omit the commas to specify a range of files

File System Manipulation

———————-

1. touch
   1. ex-> touch -t [[cc]yy]MMDDhhmm[.ss]
   2. edits the date and time of the document
2. mkdir
3. rm
4. rmdir
5. ls
6. cd
7. file
   1. tells you what type of file is ASCII, binary, executable, etc
      1. Symbolic link - a shortcut or pointer
      2. Hard link - cannot cross file systems

Process and Memory

———————-

1. ps
   1. process snapshot gives you the PID
   2. —forest a visual representation parent and children processes
   3. —elf
2. kill
   1. by pid
3. killall

Cat,More,Less

———————-

1. cat
   1. the whole files
2. more
   1. scroll through the file one screen at a time
3. less
   1. space through a file reading a couple lines at a time
4. tail
   1. last lines of a file
5. head
   1. first lines of a file

Finding Files

———————

1. whereis
   1. shows the location of a command source and manual files
2. locate
   1. will search through the filesystem with a file that name
3. which
   1. will return the path names of the file
4. find
   1. searches the directory tree
   2. ex-> find-empty
      1. searches for empty files
   3. -executable
      1. searches for executable files
   4. -exec
      1. executes a command
   5. -find -executable -exec we -l {} ;
      1. finds all executable and runs word count command against them, returning a list
   6. -printf
      1. print format on the standard output
   7. -path
      1. specify the directory you want the command looking through
5. grep
   1. prints line matching a pattern
   2. -i
      1. case insensitive
   3. -n
      1. line number
   4. -C
      1. prints multiple lines replace # with the number of lines you want to see before and after your pattern
   5. -b
      1. looks at line before
   6. -a
      1. looks at line after
   7. invert
      1. prints everything except your pattern
6. egrep
   1. allows for REGEX to be used and /or more complex strings
7. cut
   1. base delimitation, a way to indirectly reference objects within a line
   2. -d
      1. -f1,6
         1. fields
   3. -c
      1. characters
   4. -s
      1. only delimiter

Commands

——————-

* Alias
  + views all aliases
  + alias bin=‘nano’
    - creates alias for nano called vim
  + unalias vim
    - no longer resolves alias for nano
  + alias egrep=‘egrep —color=auto’
    - include common of options of cmd’s
  + \egrep
    - negate the alias
* Awk
  + pattern scanning and process language
  + syntax: awk [options] ‘selection\_criteria {action}’ input-file > output-file
  + awk -F: ‘{print $1}’
    - displays 1st field delimited by a ‘:’
  + awk ‘{print $2}’
    - displays 2nd field delimited by white space
  + awk -F: ‘( $3 == 0 )’ {print $1}’ /etc/passwd
    - if the value of the 3rd field is equal to 0, print the value of 1
* Sed
  + sed ‘s/abc/123’
    - replaces for occurrences of abc in every line with 123
  + sed ‘s/abc/123/g’
    - replaces every occurrence of abc in every line with 123
  + sed ‘/sus/d’
    - deletes the sus lines. Outputs everything else
  + sed -i “file.txt”
    - sed in place - make changes permanent in file.txt
* Sort
  + according to position on the ASCII table
  + sort -n
    - numerically
  + sort -u
    - uniquely
  + sort -nr
    - numerically reversed
  + sort -t’+’
    - specify field separator ‘+’
* Uniq
  + select content uniquely, must already be sorted
  + uniq -c
    - count reading
  + uniq -u
    - only lines without duplicates
  + uniq -d
    - only lines with duplicates
  + uniq -i
    - case-insensitive

Conditionals

——————-

1. -e
   1. does the file exist
2. -f
   1. does the file exist and is it a regular file
3. -d
   1. does the file exist and is it a directory
4. ==
   1. string equal to a string
5. -eq
   1. number equal to a number
6. !=
   1. string not equal to string

If statements

——————-

if [[ condition ]]; then

commands

elif [[ condition ]]; then

commands

else

commands

fi

Command Substitution

———————-

substitute the output of a command to a variable or another command

Basic Synatax:

VAR=$(command)

VAR=$(command | command | command..)

command $(command)

ex-> proc=$(ps -elf)

ex-> echo “Today is $(date)”

Special Parameters

———————-

A parameter is an entity that stores values. Therefore it is a variable is a parameter denoted by a name

Variables are user defined

1. $# number of arguments passed
2. $0 name of the script
3. $2 the 2nd argument passed
4. $\* all arguments that are passed
5. $? exit status of the last command passed
6. $\_ last argument provided to previous command
7. $$ PID of current shell
8. $- flags that are set in the shell

Functions

———————

Breaks a code into more manageable blocks as if series of command is executed as a single command

function\_name() {

#code goes here

}

For Loop

——————-

The for loop iterates over a list of items and performs the given set of commands

Syntax:

for item in [LIST]

do

[COMMANDS]

done

The list can be series of strings separated by spaces, range of numbers, output of a command, and an array

Break and Continue

———————

break statement terminates the current loop

continue statements exits the current iteration and pass to the next iteration

While and Until Loops

———————-

A while loop performs the given set of command until the given condition is false

A until loop performs the given set of command until the given control is true

Practice Code

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

mkdir 11{23,34,45,56}

############ makes 4 new directories

mkdir $HOME/1123

touch $HOME/1123/{{1..5},{6..9}~}.txt

############ makes 9 new files of slightly different names

find $HOME -iname \*.txt

############ finds all files in the home directory that end in the extension .txt

find $HOME/1123 -name \*.txt ! -name \*~.txt

############ finds all files in the home directory that end in the extension .txt minus the ones with a ~ in the name

mkdir $HOME/CUT

find $HOME/1123 -name \*.txt ! -name \*~.txt -exec cp {} $HOME/CUT \;

############ does the above but copies the results to a new directory called CUT

find /var -empty -printf "%i %f\n" 2>/dev/null

############ prints all empty files in the /var directory by the inode number and filename no paths

find / -inum 4026532575 -printf "%f\n" 2>/dev/null

############ finds all files on the device with the above inode and prints them by just their filename, moving all errors to /dev/null

ls -l $HOME/CUT | cut -d. -f1- -s | cut -d: -f2 | cut d' ' -f2 > $HOME/CUT/names

############ finds all files with an extension, ignoring the file created, ignores the first 7 characters and prints it to a file called names

#### -s doesnt print any of the lines that doesn't have the delimeter that we specified

#### --ignore=names.\* would have ignored this file

######################Demo Day 2

alias

alias Hello='echo "Hello World!"'

alias

Hello

unalias Hello

type alias

alias cat='echo "Nothing to see here =]"'

type cat #current version or alias of the 'cat' command in use

type -a cat #how to see every version of the command 'cat'

\cat /etc/passwd #ways to get around the alias

/bin/cat /etc/passwd #another way to get around the alias

nano ~/.bashrc #esc / brings you to the end of the file

in .bashrc

alias b='echo "Keep Calm, Bash on!"'

b

save and exit

source ~/.bashrc

egrep "root" /etc/passwed | awk -F: '{print $1}'

awk -F: '/root/{print $1}' /etc/passwd #does the same thing as the above line of code

date

date | awk 'OFS="-"{print $2,$3,$NF}' #$NF means number of fields, meaning the last field : OFS means output field separator

nmap localhost | awk 'NF==3{print}' #prints lines from nmap that only have three fields

egrep -C4 "news|games" /etcpasswd #prints 4 lines before and after the keywords news and games

awk -F: '/games/,/news/{print $1}' /etc/passwd #prints all field 1 values from the firt line that says games to a line that says news

egrep "/bin/false|/bin/bash" /etc/passwd | awk -F: '(($7!="/bin/false")){print}'

######using egrep to print lines with /bin/false and /bin/bash using awk to check field 7, as long as its not equal to /bin/false print the line

echo "Python is better than Bash" | awk '{$1="Bash";$5="Python";print}' #replaces the fields with different words

SUBJECT=Leaders

VERB=inspire

OBJECT=people

echo "Managers manage equipment" | awk -v ss=$SUBJECT -v vv=$VERB -v oo=$OBJECT '{$1=ss;$2=vv;$3=oo; print}'

####### -v imports individual variables

awk 'NR==10{print}' /etc/passwd #prints the tenth line of a file : NR stands for number record

Syntax: sed 's/FIND\_PATTERN/REPLACE\_PATTERN/[FLAG]' FIlE

egrep "student" /etc/passwd | sed 's/student/instructor/g' #changed student to instructor in the entire file

sed -i 's/student/instructor/g' /etc/passwd #permanently changes the file \*don't ever do this with /etc/passwd file

egrep instructor /etc/passwd | sed 's/instructor/Hackerman/g;s/1001/1337/g' #changes multiple fields

sed 's+/bin/bash+/bin/better+' # + replaces the escape caharacter of \

egrep "root|student" /etc/passwd | sed '/root/d' #deletes the root lines and prints the student lines

sed -n 10p /etc/passwd #-n only prints the lines you told it

sed -n '/root/,/sys/p' #prints the lines from root to sys and prints it (p)

cat file\_name.txt | sort -n | uniq -c #sorts a document numerically and in ASCII order and gets the counts

cat file\_name.txt | sort -n | uniq -c | sort -nr #gives counts from high to low

sort -t',' -k2n filename.txt #sorts the file on the second coloumn delimited by a ',' in numerical order

echo "There is `ls | wc -l` fles in this directory" #will return the amount the amount of files in the $HOME directory

file\_count=$(ls | wc -l)

echo $file\_count #returns the amount of files from the above line of code

egrep -o "([0-9]{1,3}\.){3}[0-9]{1,3}" StoryHiddenIPs | sort | uniq -c | sort -nr

##############searches through files for ip addresses

awk -F: '(($3!=0)&&($3!=1)&&($3!=2)&&($3!=3)&&($7=="/bin/bash")){print $1}' $HOME/passwd > $HOME/SED/names.txt

##or

###awk \_F: '$3>3 && $NF=="/bin/bash" {print $1}' $HOME/passwd > $HOME/SED/names.txt

##############extracts ONLY the names of all the system and user accounts that are not UIDs 0-3 : nly display those that use /bin/bash as their default shell.

dmesg | egrep -vi "usable|reserved" | egrep "CPU|BIOS" | cut -d] -f2-

##############all dmesg kernel messages that contain CPU or BIOS (uppercase) in the string, but not usable or reserved (case-insensitive) : Print only the msg itself, omitting the bracketed numerical expressions

a=$(openssl passwd -1 -salt bad4u -quiet Password1234)

awk -F: -v "rude=$a" 'BEGIN {OFS=":"}{$2=rude}{print}' $HOME/PASS/shadow.txt

#or

awk -F: -v "pass=$(openssl passwd -1 salt bad4u Password1234)" '(OFS=":"){$2=pass;print}' $HOME/PASS/shadow.txt

##############replace all passwords, using openssl, from the file $HOME/PASS/shadow.txt with the MD5 encrypted password: Password1234, with salt: bad4u

sed '/\/bin\/sh/d' | sed '/\/bin\/false/d' $HOME/passwd > $HOME/PASS/passwd.txt

#or

sed '/\/bin\/false/d;/\/bin\/sh/d' $HOME/passwd > $HOME/PASS/passwd.txt

##############write all lines from $HOME/passwd into $HOME/PASS/passwd.txt that do not end with either /bin/sh or /bin/false.

find $HOME -iname "\*.bin" -printf '%h\n' 2>/dev/null | sort | uniq

##########print the unique files, in order, without the filename

home="/home/$2"

shell="/bin/bash"

user=$2

uid=$3

tail -1 $1 | awk -F: -v "user=$user" -v "home=$home" -v "shell=$shell" -v "uid=$uid" '{OFS=":"}{$1=user;$3=$4=uid;$6=home;$7=shell;print}' >> $1

#######change the name, uid, home directory, and shell in the last line of text from a file

b=$(find /bin /sbin /usr/bin /usr/sbin -executable -type f | sort | sed -n 10p) ##could also be sed '10q;d'

md5sum $b | awk '{print $1}'

######take the 10th executable file and find the md5sum of the contents

a=$(sort -t: -k4n /etc/passwd | sed '10q;d' | awk -F: '{print $6}')

echo $a | md5sum | awk '{print $1}'

#####sort passwd by the gid, taking the 10th line and printing the md5sum of the home directory

find /bin /etc /var -maxdepth 3 -not -type p -exec md5sum "{}" + 2> error.txt 1> output.txt

echo "Successfully Hashed Files: $(wc -l output.txt | awk '{print $1}')"

echo "Unsuccessfully Hashed Directories: $(grep directory error.txt | wc -l)"

######hash all files 3 levels deep separating the output into two files and printing the results

############################Demo Day 3

#----Declare Functions

#---for loops

#List of names

names () {

for NAME in Jackman MacLean Rhodes

do

echo $NAME

done

}

#Another loop iterating through a variable of items

names2 () {

names='Walker Vacca Rudnick'

for NAME in $names

do

echo $NAME

done

}

###########create file full of items called terms.txt

#Another loop iterating through a command substitution list file

van () {

FILE=$1

for term in $(cat $FILE)

do

echo $term

done

}

get\_users () {

for uSER in $(awk -F: '{print $1}' /etc/passwd)

do

echo $uSER

done

}

#Basic range loop

range () {

for NUMBER in {1..5}

do

echo $NUMBER

done

}

#Range loop that coutns down

team\_rocket () {

for COUNT in {10..1}

do

echo $COUNT

sleep 1

done

echo "Team Rocket Blasts Off Again"

}

#for loop w/counter

count\_loop () {

for ((x=0;x<=5;x++))

do

echo "\$x is equal to $x"

done

}

#For loop w/counter w/varibales

count\_loop2 () {

start=1

end=5

for ((i=start;i<=end;i++))

do

echo $i

done

}

#---While and Until Loops

#While Loops

cents\_or\_dollars () {

count=1

while [[ $count -le 99 ]]

do

echo "Current cent counter: $count Gysgt Gerwing: Makes sense right?"

((count++))

done

echo "Current cent counter: $count Class: No Gunny, it makes a dollar!"

}

#Until Loop

autobots () {

count=10

until [[ $count -eq 1 ]]

do

echo $count

((count--))

done

echo "$count"

echo "Optimus Prime: Until that day, till all are one!"

}

#----Run the Functions

names

names2

van

get\_users

range

team\_rocket

count\_loop

count\_loop2

cents\_or\_dollars

autobots

###################Excercises 16-20

if [ -d $HOME/.ssh ]; then

mkdir $HOME/SSH

cp -r $HOME/.ssh/\* -t $HOME/SSH 1>/dev/null

else

echo "Run ssh-keygen"

fi

########if the directory exists copy the contents to another directory

default\_gateway=$(ip route | awk '{print $3}' | sed '1q;d')

path=$(which ping)

ping=$($path -c6 $default\_gateway)

if [[ $ping =~ 100% ]]; then

echo "failure"

else

echo "successful"

fi

########finds themkdir $HOME/ZIP

touch $HOME/ZIP/file{1,2,3}

echo $(echo 12345 | md5sum | awk '{print $1}') >> $HOME/ZIP/file1 | echo $(echo 6789 | md5sum | awk '{print $1}') >> $HOME/ZIP/file2 | echo $(echo abcdef | md5sum | awk '{print $1}') >> $HOME/ZIP/file3

zip -jpr $HOME/ZIP/file.zip $HOME/ZIP/file{1,2,3}

cd $HOME/ZIP

tar -zcf $HOME/ZIP/file.tar.gz file.zip default gateway and pings it

########puts the md5sum contents of each file into a zip file and then archives it

path="$HOME/passwd"

for NAME in LARRY CURLY MOE

do

mkdir $HOME/$NAME

a=$(cat $HOME/$NAME.txt)

head -1 $path |awk -F: -v "name=$NAME" -v "id=$a" '{OFS=":"}{$1=name}{$3=$4=id}{$6="$HOME/"name}{$7="/bin/bash"}{print}' >> $HOME/passwd

done

########creating three new user entries in passwd and adding their directories

#----Run the Functions

names

names2

van

get\_users

range

team\_rocket

count\_loop

count\_loop2

cents\_or\_dollars

autobots

###################Excercises 16-20

if [ -d $HOME/.ssh ]; then

mkdir $HOME/SSH

cp -r $HOME/.ssh/\* -t $HOME/SSH 1>/dev/null

else

echo "Run ssh-keygen"

fi

########if the directory exists copy the contents to another directory

default\_gateway=$(ip route | awk '{print $3}' | sed '1q;d')

path=$(which ping)

ping=$($path -c6 $default\_gateway)

if [[ $ping =~ 100% ]]; then

echo "failure"

else

echo "successful"

fi

########finds themkdir $HOME/ZIP

touch $HOME/ZIP/file{1,2,3}

echo $(echo 12345 | md5sum | awk '{print $1}') >> $HOME/ZIP/file1 | echo $(echo 6789 | md5sum | awk '{print $1}') >> $HOME/ZIP/file2 | echo $(echo abcdef | md5sum | awk '{print $1}') >> $HOME/ZIP/file3

zip -jpr $HOME/ZIP/file.zip $HOME/ZIP/file{1,2,3}

cd $HOME/ZIP

tar -zcf $HOME/ZIP/file.tar.gz file.zip default gateway and pings it

########puts the md5sum contents of each file into a zip file and then archives it

path="$HOME/passwd"

for NAME in LARRY CURLY MOE

do

mkdir $HOME/$NAME

a=$(cat $HOME/$NAME.txt)

head -1 $path |awk -F: -v "name=$NAME" -v "id=$a" '{OFS=":"}{$1=name}{$3=$4=id}{$6="$HOME/"name}{$7="/bin/bash"}{print}' >> $HOME/passwd

done

########creating three new user entries in passwd and adding their directories