Print working directory – pwd – where will execute - /User/Munks

ls – what’s in the directory – files and folders

ls foldername

ls –l foldername to see different between files and folders

d- directory

* file

man ls – manual pages -permissions

mkdir foldername – make a directory

rmdir foldername – remove directory

clear to clean screen

cd – to change directory

cp – copy directory

cp filename newfilename

rm filename – remove filename

cat – see what’s inside a file

* stick files together
* show lines on a screen – will only see if it’s text

more – add more tab

head – shows first copy lines

tail – shows last lines

Expansions:

~ home

touch {apple, banana, cherry, durian} – create files, each

touch file\_{1..1000} – creates new 1000 files

touch file\_{01..1000} – zero padding

echo {1..10..2} – get numbers counting by 2 – can use with letters – intervals – can’t use on my version

piping – takes the result of one command and sends into another

ls | more – ls into more – pagination

grep – search files for patters of text – search through log files

ls filename

search for username: grep searchterm filename

grep --color=auto searchterm filename

export GREP=OPTIONS=’--color=auto’

grep –i break-in filename – breakin attempts

grep searchterm filename | awk {‘print $12’} – print 12th item

sed

awk

pull specific information

ping example.com

Cntl C to stop

Ping –c 1 example.com

Response time: Ping –c 1 example.com | grep searchterm | cut –d(delimiter) = -f (field) number of item

BASH – whole interactive shell

Bash script – contains variables, arguments, logic

#!/bin/bash

hashbang/ shebang – Path to executable

# comment – ignores text

#! – bash script

Script:

Nano my.sh – shell script

Echo statement – prints

No quotes – interpret as it finds thing—special characters /

Single quotes – comes back as a string

Double quotes – uses both

Variables:

Must start with a letter

#!/bin/bash

#This is a basic bash script.

a=Hello

b="Good morning!"

c= 16

echo $a

echo $b

echo $c

echo "$a, $b I have $c apples."

chmod +x my.sh

Run with ./my.sh

declare –i d=123 -- # d is an integer

declare -r e=456 -- #e is read-only - can't not be manipulated

declare -l f="LOLCats" -- # f is lolcats

declare -u g="uppercase" == # g is UPPERCASE

echo $pwd - returns current directory

echo $MACHTYPE - Machine type

echo $HOSTNAME - returns system name

echo$BASH\_VERISON - bash versions

echo$SECONDS - number of sessions a bash sessions has been open

echo $) - name of script

Internal variables

d=$(pwd)

echo d

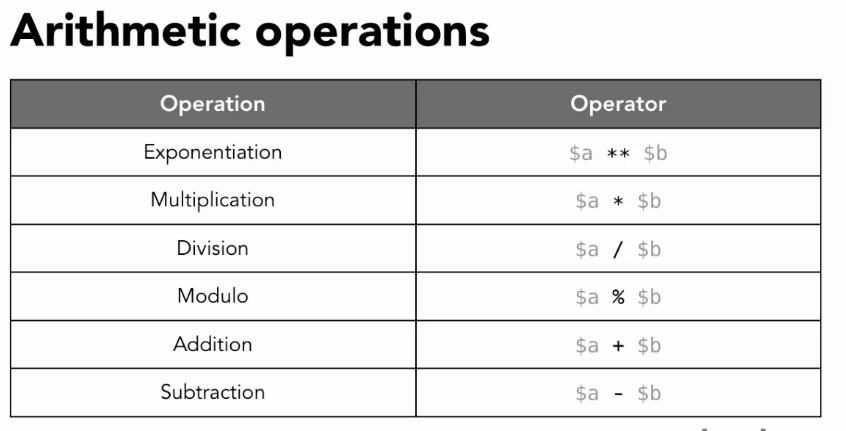
runs command setting to variable d

a=$(ping -c 1 michellemunksgard.com | grep 'bytes from' | cut -d = -f 4)

=response from a server

create in script to use $a later

Arithmetic operations--



(( )) - Double parenthesis to avoid cat

d=2

e=$((d+2))

echo $e

#!/bin/bash

d=2

e=$((d+2))

echo $e

((e++))

echo $e

((e--))

echo $e

echo

((e+=5))

echo $e

((e\*=3))

echo $e

((e/=3))

echo $e

((e-=5))

echo $e

Michelles-iMac:~ Munk$ ./my.sh

4

5

4

9

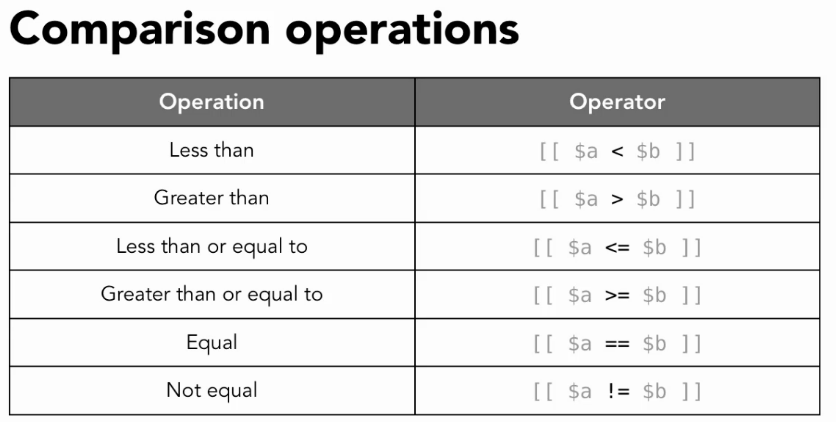
27

9

4

man bc

Comparison operations:

[[ ]] 

[[ "cat" == "cat" ]]

echo $?

[[ "cat" == "dog" ]]

echo $?

Results:

0 - success

1 - failure

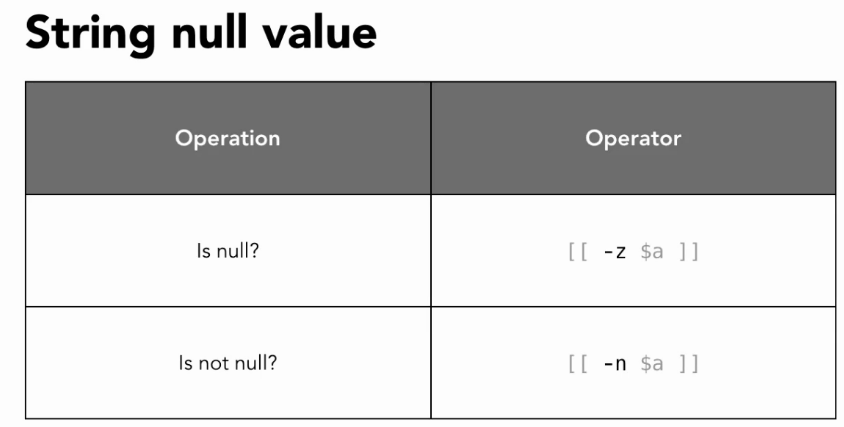
[[ 20 > 100 ]]

should be

[[ 20 -gt 100 ]]

Compare as numbers.

Integers



&& -- if both are correct

echo ${#variable} - how many characters.

d=${c:3} - show characters after the third

e=${c:3:4} = Asks for 4 characters after the third

echo ${c: -4:3} - count from end of string

Replacing

fruit="apple banana banana cherry"

echo ${fruit/banana/durian}

result: apple durian banana cherry

echo ${fruit//banana/durian}

result: apple durian durian cherry

echo ${fruit/#apple/durian} - will change if it's at the beginning of the string

result: durian banana banana cherry

echo ${fruit/%cherry/durian} - will change if it's at the end of the string

result: apple banana banana durian

- if the word is not found at the beginning or end it will leave it unchanged.

echo ${fruit/c\*/durian} - matching

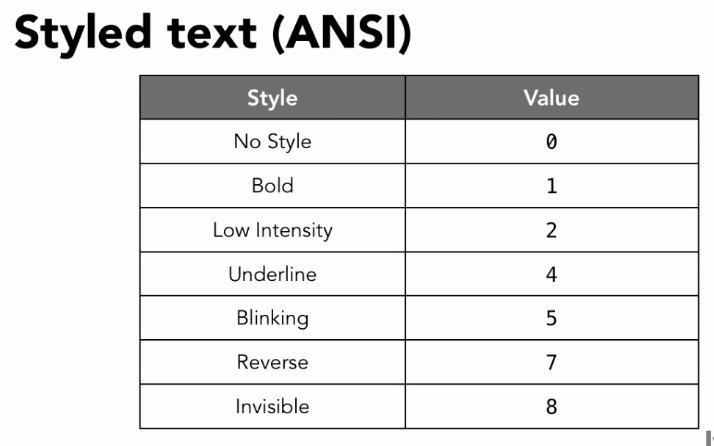
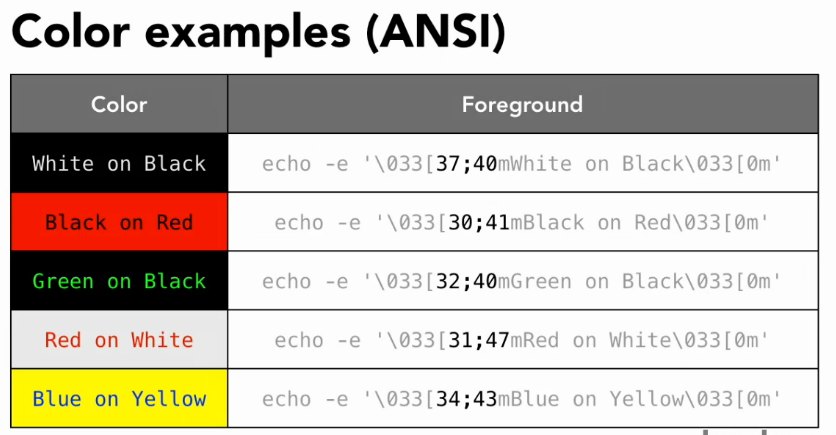
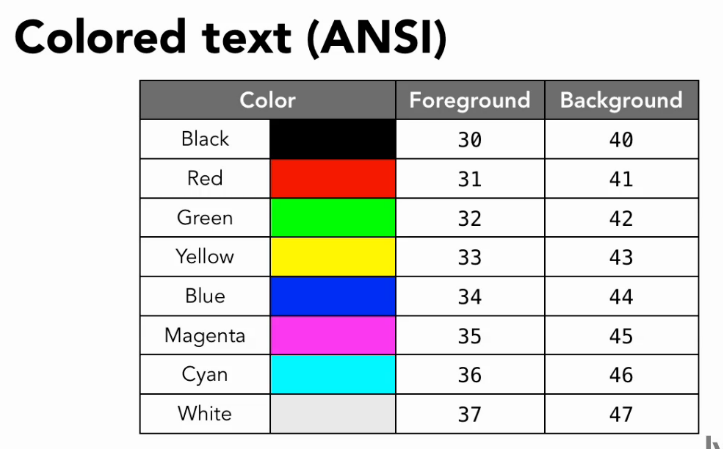
result: apple banana banana durian

Colored Text:

escaped sequence:

echo -e "\033[34;42mColor Text\033[0m'

34;42 - blue/ green



\033[0m - clear out

Break down and use as variables.

#!/bin/bash

flashingred = "\033[5;31;40m"

red="\033[31;40m"

none="\033[0m"

echo -e $flashingred"ERROR: "$none$red"Something went wrong."$none

