I. Conditionals / Input

1.

#!/bin/bash

# You can use read to receive input which is stored in name

# The p option says that we want to prompt with a string

read -p "What is your name? " name

echo "Hello $name"

read -p "How old are you? " age

# You place your condition with in []

# Include a space after [ and before ]

# Integer Comparisons: eq, ne, le, lt, ge, gt

if [ $age -ge 16 ]

then

echo "You can drive"

# Check another condition

elif [ $age -eq 15 ]

then

echo "You can drive next year"

# Executed by default

else

echo "You can't drive"

# Closes the if statement

fi

2. Extended integer test

#!/bin/bash

read -p "Enter a number : " num

if ((num == 10)); then

echo "Your number equals 10"

fi

if ((num > 10)); then

echo "It is greater then 10"

else

echo "It is less then 10"

fi

if (( ((num % 2)) == 0 )); then

echo " It is even"

fi

# You can use logical operators like &&, || and !

if (( ((num > 0)) && ((num < 11)) )); then

echo "$num is between 1 and 10"

fi

# && and || can be used as control structures

# Create a file and then if that worked open it in Vim

touch samp\_file && vim samp\_file

# If samp\_dir doesn't exist make it

[ -d samp\_dir ] || mkdir samp\_dir

# Delete file rm samp\_file

# Delete directory rmdir samp\_dir

3. Testing strings

#!/bin/bash

str1=""

str2="Sad"

str3="Happy"

# Test if a string is null

if [ "$str1" ]; then

echo "$str1 is not null"

fi

if [ -z "$str1" ]; then

echo "str1 has no value"

fi

# Check for equality

if [ "$str2" == "$str3" ]; then

echo "$str2 equals $str3"

elif [ "$str2" != "$str3" ]; then

echo "$str2 is not equal to $str3"

fi

if [ "$str2" > "$str3" ]; then

echo "$str2 is greater then $str3"

elif [ "$str2" < "$str3" ]; then

echo "$str2 is less then $str3"

fi

# Check the file test\_file1 and test\_file2

file1="./test\_file1"

file2="./test\_file2"

if [ -e "$file1" ]; then

echo "$file1 exists"

if [ -f "$file1" ]; then

echo "$file1 is a normal file"

fi

if [ -r "$file1" ]; then

echo "$file1 is readable"

fi

if [ -w "$file1" ]; then

echo "$file1 is writable"

fi

if [ -x "$file1" ]; then

echo "$file1 is executable"

fi

if [ -d "$file1" ]; then

echo "$file1 is a directory"

fi

if [ -L "$file1" ]; then

echo "$file1 is a symbolic link"

fi

if [ -p "$file1" ]; then

echo "$file1 is a named pipe"

fi

if [ -S "$file1" ]; then

echo "$file1 is a network socket"

fi

if [ -G "$file1" ]; then

echo "$file1 is owned by the group"

fi

if [ -O "$file1" ]; then

echo "$file1 is owned by the userid"

fi

fi

4. Use case to when it makes more sense then if

#!/bin/bash

read -p "How old are you : " age

# Check the value of age

case $age in

# Match numbers 0 - 4

[0-4])

echo "To young for school"

;; # Stop checking further

# Match only 5

5)

echo "Go to kindergarten"

;;

# Check 6 - 18

[6-9]|1[0-8])

grade=$((age-5))

echo "Go to grade $grade"

;;

# Default action

\*)

echo "You are to old for school"

;;

esac # End case

II. Looping

1. While Loop

#!/bin/bash

num=1

while [ $num -le 10 ]; do

echo $num

num=$((num + 1))

done

2. Continue and Break

#!/bin/bash

num=1

while [ $num -le 20 ]; do

# Don't print evens

if (( ((num % 2)) == 0 )); then

num=$((num + 1))

continue

fi

# Jump out of the loop with break

if ((num >= 15)); then

break

fi

echo $num

num=$((num + 1))

done

3. Until loops until the loop is true

#!/bin/bash

num=1

until [ $num -gt 10 ]; do

echo $num

num=$((num + 1))

done

4. There are many for loop options. Here is the C form.

#!/bin/bash

for (( i=0; i <= 10; i=i+1 )); do

echo $i

done

5. We can cycle through ranges

#!/bin/bash

for i in {A..Z}; do

echo $i

done

III. Positional Parameters

1. Positional parameters are variables that can store data on the command line in variable names 0 - 9

a. $0 always contains the path to the executed script

b. You can access names past 9 by using parameter expansion like this ${10}

2. Add all numbers on the command line

#!/bin/bash

# Print the first argument

echo "1st Argument : $1"

sum=0

# $# tells you the number of arguments

while [[ $# -gt 0 ]]; do

# Get the first argument

num=$1

sum=$((sum + num))

# shift moves the value of $2 into $1 until none are left

# The value of $# decrements as well

shift

done

echo "Sum : $sum"